KU LEUVEN

FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN

TECHNOLOGIECAMPUS GENT

Improving evolutionary algorithms for multi-objective optimisation

Generating high trade-off solutions and introducing a novel stopping criterion

Viviane DE BUCK

Promotor: Jan Van Impe

Co-promotoren: Ihab Hashem
Carlos André Muñoz Lopez
Philippe Nimmegeers

Masterproef ingediend tot het behalen van de graad van master of Science in de industriële wetenschappen: Biochemie

Academiejaar 2017 - 2018

Acknowledgements

This thesis has presented me with a big challenge by getting me out of my comfort zone. This was a deliberate choice, but it would have been an impossible one without the support of many.

I would firstly like to thank my supervisor, Professor Jan Van Impe, for providing me with the opportunity to study and explore the presented subject. I would also like to thank Ihab, Philippe, and André for their guidance and support on a daily basis, but also the freedom and trust they have given me during the process. Additionally, their quick and thorough reviews of the drafts of this text are highly appreciated and allowed me to improve it.

On a more personal note, I would like to thank my parents and family for their care and encouragement, and this not only during this last semester. My gratitude also goes out to my grandparents, for the wisdom they have imparted to me, as well to those grandparents who are no longer with us but whose remembrances have since been an additional motivation. Our family cat, Helmer, also deserves a special mention for always keeping me company and being a little bundle of joy and love.

Lastly I would like to thank my friends for filling these past four years with laughter, friendship and unforgettable memories.

Viviane De Buck,
June 2018.

Abstract

Contemporary developments in the perception of our environment have caused a completely new process industry. Production processes now have to operate as optimal as possible with respect to different, often conflicting, objectives as for instance optimality with respect to societal, environmental and economical aspects. In such case, no one optimal solution exists and decision makers need to resort to trade-off (or Pareto optimal) solutions. The problem of optimising a process with respect to different conflicting objectives, is called a multi-objective optimisation problem (MOOP). MOOPs have gained a lot of attention in different applications throughout the past decade. MOOPs are however mathematically challenging problems and are generally solved via the use of dedicated algorithms. The two major algorithm categories are deterministic and stochastic algorithms. The former converts the MOOP in a set parameterised single-objective optimisation problems (SOOPs), while the latter tackles the MOOP in its entirety. The focus of this thesis is on the stochastic algorithms, and more specifically on the evolutionary algorithms Non-dominated sorting genetic algorithm-II (NSGA-II) and Non-dominated sorting genetic algorithm-III (NSGA-III). These algorithms are widely acclaimed but still show two major shortcomings: (i) they are incapable of distinguishing between solutions based on their trade-off and distribution; (ii) they utilise a problem-irrelevant stopping criterion (i.e. reaching a pre-defined number of iterations). The lack of a trade-off based selection procedure results in a Pareto front that contains solutions that present no relevant information to the user. The flawed stopping criterion results in an unnecessary high computation time. To alleviate these shortcomings, the tM- and eventually tDOM-algorithms are developed. These algorithms can distinguish between solutions based on their trade-off via the implementation of t-domination. The tM-algorithm furthermore uses the convergence of solutions as a stopping criterion, via an additional user defined parameter. The tDOM-algorithms stop if all the solutions of two subsequently generated solution populations do not t-dominate each other. Both novel algorithms are capable of generating a Pareto front with a trade-off based solution resolution and display a significant gain in computation time.

Key words: Multi-objective optimisation problems, NSGA-II, NSGA-III, Evolutionary algorithms, Trade-off, t-domination

Samenvatting

Hedendaagse ontwikkelingen in de perceptie van ons milieu hebben een compleet nieuwe procesindustrie veroorzaakt. Productieprocessen moeten nu zo optimaal mogelijk werken met betrekking tot verschillende, vaak tegenstrijdige, doelstellingen op vlak van maatschappelijke, milieu- en economische aspecten. In dergelijke gevallen bestaat er geen optimale oplossing en moeten beslissers hun toevlucht nemen tot trade-off (of Pareto-optimale) oplossingen. Het optimaliseren van een proces met betrekking tot verschillende tegenstrijdige doelen (of objectieven), wordt een multi-objectief optimalisatieprobleem (MOOP) genoemd. MOOP's hebben het afgelopen decennium veel aandacht gewonnen in verschillende toepassingsgebieden. MOOP's zijn echter wiskundig uitdagende problemen en worden in het algemeen opgelost door middel van speciale algoritmen. De twee belangrijkste algoritmecategorieën zijn deterministische en stochastische algoritmen. Terwijl de eerstgenoemde de MOOP converteert in een reeks optimaliseringsproblemen met een enkel objectief (SOOP), behandelt de laatste de MOOP in zijn geheel. Deze scriptie focust op de stochastische algoritmen en meer specifiek op de evolutionaire algoritmen Non-dominated sorting genetic algorithm-II (NSGA-II) en Non-dominated sorting genetic algorithm-III (NSGA-III). Deze algoritmen worden alom geprezen, maar vertonen twee belangrijke tekortkomingen: (i) ze zijn niet in staat onderscheid te maken tussen oplossingen op basis van hun afweging; (ii) ze gebruiken een probleem-irrelevant stopcriterium (nl. het bereiken van een vooraf gedefinieerd aantal iteraties). Het ontbreken van een selectieprocedure gebaseerd op de afweging van een oplossing, resulteert in een Pareto front dat oplossingen bevat die voor de gebruiker geen relevante informatie bevatten. Het gebrekkige stopcriterium resulteert in een onnodige hoge rekentijd. Om deze tekortkomingen te verhelpen, zijn de tM-, en uiteindelijk tDOM-algoritmen ontwikkeld. Deze algoritmen kunnen een onderscheid maken tussen oplossingen op basis van hun afweging via de implementatie van t-dominatie. Het tM-algoritme gebruikt als stopcriterium de convergentie van de oplossingen via een extra, door de gebruiker gedefinieerde parameter. De tDOM-algoritmen stoppen als alle oplossingen van twee achtereenvolgend gegenereerd oplossingsverzamelingen elkaar niet t-domineren. Beide nieuwe algoritmen zijn in staat om een Pareto front te genereren met een afweging-gebaseerde oplossingsresolutie, en vertonen simultaan een significante winst in rekentijd.

Sleutelwoorden: Multi-objectieve optimalisatie problemen, NSGA-II, NSGA-III, Evolutionaire algoritmes, Trade-off, t-dominantie

Wetenschappelijke samenvatting

Sinds enkele decennia heeft een compleet nieuwe en duurzame procesindustrie zich steeds prominenter geprofileerd. Door het toenemende besef dat klimaatverandering wel degelijk een acuut probleem vormt, wordt er steeds naar nieuwe manieren gezocht om energie te besparen en minder broeikasgassen uit te stoten. Niet alleen overheden staan hierdoor voor een aanzienlijke uitdaging, maar ook de industrie moet de handen uit de mouwen steken. Waar het vroeger voldoende was dat industriële processen functioneel waren en een relatief hoge opbrengst leverden, moeten ze nu ook duurzaam en energiezuinig zijn. Deze extra eisen zorgen ervoor dat heel veel processen moeten geoptimaliseerd worden.

Aangezien productieprocessen, uit welke industrietak ook, vaak complex zijn, is hun optimalisatie derhalve ook niet eenduidig. Zo moet er aan verschillende eisen simultaan voldaan worden, hoewel dit vaak tot tegenstrijdige resultaten leidt. Wanneer bv. een chemisch proces beschouwd wordt, zal een energiebezuiniging in de vorm van een verlaging van de reactortemperatuur vaak ook aanleiding geven tot een verlaging in opbrengst. Terwijl er dus in een bepaald objectief gewonnen wordt (i.e. energie besparen), wordt in een ander objectief verloren (i.e. opbrengst). Dergelijke optimalisatieproblemen waarbij simultaan aan verschillende eisen of objectieven moet worden voldaan, worden multi-objectieve optimalisatie problemen (MOOP) genoemd. Een MOOP heeft niet één, maar oneindig veel optimale oplossingen. Het hoeft niet te verwonderen dat, indien het proces dat geoptimaliseerd moet worden op zich reeds complex is, de mathematische beschrijving ervan in de vorm van een MOOP nog veel complexer is. Daarom wordt meestal gebruik gemaakt van (computer-)algoritmes voor het minimaliseren van een MOOP.

Er bestaan twee hoofdcategorieën onder deze computeralgoritmes: deterministische algoritmes en stochastische algoritmes. De eerstgenoemde algoritmes vormen een MOOP om tot een reeks van optimalisatieproblemen met maar een enkel objectief (SOOP) door middel van parameters. Deze SOOP worden vervolgens individueel opgelost waardoor een oplossing van de initiële MOOP verkregen wordt. Deze oplossingen worden ook wel Pareto-optimaal of niet-gedomineerd genoemd. De verzameling van al deze oplossing wordt het Pareto front genoemd. Ondanks dat deterministische algoritmes zeer populair zijn, hebben ze ook enkele niet onmiskenbare tekortkomingen. Zo kan het feit dat ze maar één oplossing per iteratie kunnen genereren als een nadeel gezien worden. Deterministische algoritmes zijn ook meer gevoelig om naar lokale optima te convergeren. Bovendien zorgt het feit dat een groot aantal SOOP's moeten opgelost worden ervoor dat de algoritmes vaak tijdrovend en complex zijn.

De hoofdfocus van deze scriptie ligt echter bij de stochastische algoritmes. Dit type algoritmes behandelt de MOOP in zijn integriteit en is in staat meerdere Pareto-optimale oplossingen te genereren per iteratie. Evolutionaire algoritmes vormen een subcategorie van de stochastische algoritmes. Non-dominated sorting genetic algorithm-II (NSGA-II) en non-dominated sorting genetic algorithm-III (NSGA-III) zijn veruit de meest gekende evolutionaire algoritmes. De algemene werkmethode van dit type algoritmes is gebaseerd op seksuele voortplanting en selectie, net zoals in de natuur. Een random initiële populatie van MOOP oplossingen wordt gebruikt om nieuwe oplossingen te genereren. Door telkens de beste (meest optimale)

oplossingen te selecteren voor verdere reproductiestappen, convergeren de oplossingen uiteindelijk naar het Pareto front.

Hoewel NSGA-II en NSGA-III alle lof verdienen, vertonen ze echter wel twee fundamentele tekortkomingen. Zo zijn deze algoritmes niet in staat een onderscheid te maken tussen de gegenereerde oplossingen op basis van de afweging van de oplossingen in kwestie. Oplossingen met een hogere afweging worden namelijk geprefereerd door de beslissingsmaker. Een tweede tekortkoming van de NSGA-II en NSGA-III algoritmes is hun gebruik van een probleem-irrelevant stopcriterium. De algoritmes worden enkel onderbroken wanneer een bepaald aantal iteraties bereikt is. Dit arbitraire stopcriterium, welke gedefinieerd moet worden door de gebruiker, zorgt er vaak voor dat dure rekentijd onnodig verbruikt wordt. Aangezien zowel NSGA-II als NSGA-III elitaire algoritmes zijn, zullen oplossingen zich niet terug verwijderen van het Pareto front eenmaal ze volledig geconvergeerd zijn. De beste oplossingen van de vorige oplossingsgeneratie worden namelijk onveranderd overgenomen in de huidige oplossingsge-neratie. Het voortzetten van de algoritmes nadat dit punt bereikt is, is dus nutteloos.

De bovenstaande tekortkomingen hebben in eerste instantie aanleiding gegeven tot de ontwikkeling van de nieuwe tM-algoritmes. Deze algoritmes zijn in staat om oplossingen met een hoge afweging te beklemtonen en beschikken over een probleem-relevant stopcriterium. Dit resulteert in de generatie van Pareto fronten met een duidelijke, afweging gerelateerde oplossingsresolutie enerzijds, en anderzijds een significante daling in de vereiste rekentijd. Oplossingen met een hoge afweging worden onderscheiden van die met een lage afweging door middel van het aantal oplossingen die zich in hun regio's van praktisch onbetekenende afweging (PIT-regio) bevinden. De PIT-regio werd geïntroduceerd door Mattson et al. (2014) en is een kruisvormige ruimtelijke eenheid met in het centrum de oplossing in kwestie. De PIT-regio van een oplossing wordt gedefinieerd door enerzijds de afweging Δt en anderzijds de distributie Δr. Deze twee parameters worden door de gebruiker aan het systeem opgelegd en zijn gebaseerd op minutieus onderzoek. De zogenoemde afwegingsteller wordt in de tM- en tDOM-algoritmes gebruikt om aan te tonen hoeveel andere oplossingen er zich in de PIT-regio van een bepaalde oplossing bevinden. Indien de oplossing in stijgende volgorde gesorteerd worden op basis van hun afwegingsteller, zullen de oplossingen met een hoge afweging in de hogere posities van de oplossingsverzameling gesorteerd worden.

Hoewel het stopcriterium van de tM-algoritmes de algoritmes in staat stelde om veel rekentijd te besparen, werd er nog steeds een grote tekortkoming waargenomen. Het stopcriterium van de tM-algoritmes was gebaseerd op de convergentie van de oplossingen. De mean ideal distance (MID) werd als prestatieparameter gebruikt om de convergentie van de oplossingen te kwantificeren. Deze parameter stelt namelijk de genormaliseerde gemiddelde euclidische afstand voor tussen de genereerde oplossingen enerzijds en het Utopia punt (i.e. het punt dat alle individuele minima van de beschouwde objectieven bevat) anderzijds. Aangezien NSGA-II en NSGA-III elitaire algoritmes zijn, kan aangenomen worden dat indien de convergentie van de oplossingen niet verbetert, of stagneert, de oplossing hoogstwaarschijnlijk het Pareto front bereikt hebben. Deze stagnatie in convergentie kan waargenomen worden in de nagenoeg identieke MID-waarde van twee opeenvolgende oplossingspopulaties. Het tM-algoritme stopt wanneer het verschil in MID tussen twee opeenvolgende oplossingspopulaties kleiner wordt dan een tolerantiewaarde ΔMID, welke door de gebruiker dient gedefinieerd te worden. Aangezien deze laatste echter vaak geen idee heeft wat een goede convergentie is voor zijn/haar beschouwd optimalisatieprobleem, zal de ΔMID waarde opnieuw arbitrair vastgelegd worden.

De gebruiker kan echter wel gefundeerde waarden opgeven voor de reeds vernoemde afweging Δt en distributie-parameters Δr. Er kan bovendien gesteld worden dat indien een algoritme in staat is actief die oplossingen te selecteren die de hoogste diversiteit hebben en algemeen de hoogste toegevoegde

waarde aan de oplossingspopulatie bieden, dat algoritme ook in staat is vast te stellen wanneer de oplossingen dat het genereert niet langer aan de eisen van de gebruiker voldoen. Het stopcriterium van de tDOM-algoritmes is hierop gebaseerd. Deze algoritmes worden stopgezet indien alle oplossingen van twee opeenvolgend gegenereerde oplossingspopulaties zich in elkaar PIT-regio bevinden. De oplossingen van die twee populaties worden door de gebruiker namelijk niet langer als verschillend beschouwd. Aangezien NSGA-II en NSGA-III elitaire algoritmes zijn, zal dit scenario zich enkel voordoen indien de oplossingen voldoende convergeert zijn naar het Pareto front. Het stopcriterium van de tDOM-algoritmes vereist dus geen additionele parameter, maar maakt gebruikt van gefundeerde eisen opgelegd door de gebruiker. Merk op dat de in deze thesis ontwikkelde algoritmes gebaseerd zijn op fundamenteel onderzoek. Doordat ze niet specifiek geworteld zijn in een bepaalde ingenieurstak, zijn ze veelzijdig en toepasbaar in verscheidene ingenieurstoepassingen.

Contents

List of Symbols

$\mathbf{a}$	Lower boundary
$\mathbf{b}$	Upper boundary
C	(Constrained) solution space
d	Amount of subdivisions alongside each objective
$\mathbf{F}$	Objective vector
$FC(\mathbf{x})$	Feasibility counter of solution $\mathbf{x}$
$\mathbf{g}(\mathbf{x})$	Inequality constraints
H	Number of reference points
$\mathbf{h}(\mathbf{x})$	Equality constraints
M	Number of objectives
N	Size of the solution population
N_{nd}	Number of non-dominated solutions
n	Number of process variables
ne	Number of equality constraints
ni	Number of inequality constraints
t	Trade-off

List of Acronyms

CHIM	Convex hull of individual minima
DM	Decision maker
EA	Evolutionary algorithm
(E)NNC	(Enhanced) Normalised normal constraint method
FPOS	Fraction of Pareto-optimal solutions
MID	Mean ideal distance
MOOP	Multi-objective optimisation problem
NBI	Normal boundary intersection method
NDF	Non-dominated front
NSGA-II	Non-dominated sorting genetic algorithm II
NSGA-III	Non-dominated sorting genetic algorithm III
PF	Pareto front
PIT	Practically insignificant trade-off
SNDS	Spread of non-dominated solutions
SOOP	Single-objective optimisation problem
WS	Weighted sum method

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction and positioning the thesis topic

Background

The biochemical and chemical industry is a strong economical player and employer in Belgium. In 2015, the chemical, plastics, and life science industry in Belgium was responsible for a turnover of more than 64 billion Euro, 90,000 direct employees and 150,000 indirect jobs. Almost one third of the total manufacturing added value of 2015 in Belgium was derived from chemical industry (CEFIC, 2017).

Although delivering multiple job opportunities and great turnover figures, the chemical industry faces ever-increasing challenges. The chemical industry is known to be a potential big polluter and in the context of climate change and resource shortages, regulations are becoming increasingly strict. Companies are therefore looking for new ways to save on energy, resources and money. When optimising a (bio-)chemical process, often multiple objectives must be met simultaneously. Because these objectives are commonly determined by several independent variables and have contradictory optima, solving a multi-objective optimisation problem does not yield one optimal solution but an infinite set of optimal solutions. These solutions are located on the so-called Pareto front (Vallerio et al., 2015).

In order to generate a Pareto front with a sufficient solution resolution, convergence and spread, various solving methods have been developed. A major category within these solving methods, are the evolutionary algorithms. One of their main features is that they are capable of generating multiple Pareto-optimal solutions (i.e solutions located on the Pareto front) in a single run. This makes them an excellent choice to solve multi-objective optimisation problems, especially if the decision maker is interested in a diverse set of solutions.

Thesis goal and objectives

Early evolutionary algorithms were lacking elitism and had a high computational complexity. Elitism allows the algorithm to keep the best solutions of the previous iteration unchanged in the current one, which significantly increases the convergence speed of the algorithm. In order to maintain a decent spread in solutions, the user had to define a sharing parameter. However, the overall efficiency of the algorithm was highly dependent on the value of this sharing parameter. The lack of elitism of the first evolutionary algorithms on the other hand prevented a fast convergence to the Pareto front.

More contemporary evolutionary algorithms like non-dominated sorting genetic algorithm II (NSGA-II) resolved the shortcomings of the early algorithms (Deb et al., 2002). Non-dominated sorting genetic algorithm III (NSGA-III) is based on the framework of NSGA-II, but was especially developed to handle multi-objective problems with four or more objectives (Deb and Jain, 2014). Although these algorithms were a big step in the right direction, several improvements can still be done.

The overall goal of this thesis is therefore to point out the shortcomings of the currently available NSGA-II and NSGA-III algorithms and to resolve these shortcomings. The focus will be laid on the spread of solutions on the Pareto front, and more specifically in the high trade-off regions of the Pareto front. A solution has a high trade-off if a small change in the value of one objective (or in the corresponding process variables) has a big influence on the values of other objectives. One of the main targets of this thesis will be to enable evolutionary algorithms to distinguish low trade-off solutions from high trade-off solutions and subsequently favouring the latter ones. This will eventually lead to the development of novel genetic algorithms that take the trade-off a generated solution into account.

Additionally, the stopping criteria of the currently available NSGA-II and NSGA-III algorithms are arbitrary chosen and have little relevance to the main goal of the algorithms, i.e. generating Pareto-optimal solutions. The possibilities to improve these stopping criteria will be further examined and will be implemented in the novel algorithms.

Thesis outline

In the literature study a distinction is made between multi-objective optimisation and evolutionary algorithms. In the first chapter of this part, the concept of multi-objective optimisation is explained together with three widely used scalarisation solving methods (Weighted sum, Normal boundary intersection, and (enhanced) Normalised normal constraint method).

The following chapter focuses on evolutionary algorithms. Firstly their advantages over the previously described scalarisation solving methods will be emphasised together with other general considerations. Subsequently NSGA-II and NSGA-III are thoroughly discussed. A constraint-handling scheme is presented along the presentation of both algorithms.

In the following part, the materials and methods used during the course of this thesis are discussed. The used case studies are also presented in this part as well as the used software and source codes.

The remaining part of the thesis is dedicated to the representation and discussion of the made observations and improvements. The first chapter of this part focusses on enabling both the NSGA-II and NSGA-III algorithms to handle constrained multi-objective problems. Two feasibility checks are introduced and the obtained so-called constrained-algorithms are tested on the case studies. The case studies enable simultaneously to point out the shortcomings of the original NSGA-II and NSGA-III algorithms.

In the subsequent chapter, these shortcomings are further examined, leading up to the proposal of a novel tM-evolutionary algorithm. This algorithm is based on the NSGA-II and NSGA-III algorithms but uses a problem-relevant stopping criterion and is capable in distinguishing between solutions based on their trade-off. The tM-algorithms will likewise be tested on the numerical case studies. Based on the results of these case studies, a more advanced and final novel algorithm is proposed: the tDOM-algorithm.

The tDOM-algorithms are discussed in detail in Chapter 7. These algorithms use, additionally to non-dominated sorting, a t-non-dominated sorting algorithm. t-domination is a variant of non-domination and allows the algorithm to pinpoint when it is suitable to stop, based on the trade-off of the generated solutions. Because the trade-off of solutions is a property that is imposed by the user, based on underpinned arguments, the tDOM-algorithms use in fact a parameterless and problem-relevant stopping criterion. This, jointly with the trade-off based Pareto front solution resolution, makes the tDOM-algorithms very promising in view of further use and implementation in commercial software.

The last part of this thesis focuses on the formulation of a conclusion of the made observations and obtained results. Some perspectives for further research are provided simultaneously.

Part II

Literature study

Chapter 2

Multi-Objective Optimisation Problems

2.1 Introduction

In a world of increased awareness of climate change and resource depletion, it is not surprising that chemical production companies, and other companies in general, are becoming more interested in optimising their workflow and processes. Using optimised processes enables companies to simultaneously increase their yield and save energy, resources, and money.

The optimisation of, for instance, a fed-batch reactor consists of multiple objectives which all have to be considered at the same time. The goal is to optimise the separate objectives whilst keeping in mind that the (optimal) values of the individual objectives are often linked and dependent on each other. In the case of the fed-batch reactor, the feed rate of reagents, the temperature of the reactor, the vigorousness of the stirrer, etc. all have an influence on the overall yield of the reactor and its energy demand (Vallerio et al., 2015).

If it is possible to identify the independent process variables that influence the values of the considered objectives and if these objectives are translatable into mathematical equations, a multi-objective optimisation problem (MOOP) can be formulated. The convention is that an optimisation problem is mathematically formulated as a minimising problem, despite certain objectives might have optimal maximum values (for instance the process yield). Objectives that have to be maximised are however easily converted to objectives that have to be minimised by multiplying the objective function in question by -1.

In this chapter, a multi-objective optimisation problem will be mathematically described. Subsequently the two major solving strategies will be explained, being scalarisation methods and vectorisation methods. At the end of this chapter, three commonly used scalarisation methods are introduced.

2.2 Mathematical description of a multi-objective optimisation problem

In general it is possible to describe an optimisation problem mathematically as follows (Das and Dennis, 1997):

$$\min_{x \in C} F(x) \tag{2.1}$$

Equation (2.1) is a scalar equation: F stands for a certain goal that needs to be minimised (e.g. cost). x is the optimisation variable or process variable, in this case a scalar (e.g. time). The process variable is often subjected to constraints. The constraints render a feasible region C. Only process variables that are part of C will provide a feasible solution of the optimisation problem.

On a more general note however, it is more likely that an optimisation problem consists out of multiple

goals that need to be optimised at the same time and that can have contradictory optima. In this scenario, Equation (2.1) becomes (Das and Dennis, 1997):

$$\min_{\mathbf{x}\in C} F(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_M(\mathbf{x}) \end{bmatrix} \tag{2.2}$$

with:

$$C = \{\mathbf{x} : \ \mathbf{h}(\mathbf{x}) = 0; \ (\mathbf{g}(\mathbf{x}) \leq 0; \ \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}\} \tag{2.3}$$

and

$$F : \mathbb{R}^n \mapsto \mathbb{R}^M \tag{2.4}$$

$$h : \mathbb{R}^n \mapsto \mathbb{R}^{ne} \tag{2.5}$$

$$g : \mathbb{R}^n \mapsto \mathbb{R}^{ni} \tag{2.6}$$

A solution $\mathbf{x}^*$ of the MOOP is Pareto-optimal or non-dominated if there exist no other solution $\mathbf{x}$ for which $F(\mathbf{x}) \leq F(\mathbf{x}^*)$ and additionally for this solution $\mathbf{x}$ also at least one of the objectives $f_k(\mathbf{x}) < f_k(\mathbf{x}^*)$ with $k \in \{1 \ldots M\}$. If there exist such a solution $\mathbf{x}$, then $\mathbf{x}^*$ is not Pareto-optimal. M stands for the number of objectives, n is the number of decision variables, ne is the number of equality constraints and ni is the number of inequality constraints (Das and Dennis, 1997). The Pareto front of an optimisation problem is the visual representation of a set of Pareto-optimal solutions.

Equation (2.3) defines the solution space C. Every possible solution $\mathbf{x}$ of the MOOP is part of this set and is subjected to several equality constraints (Equation (2.5)), inequality constraints (Equation (2.6)) and boundaries $\mathbf{a}$ and $\mathbf{b}$, which represent the lower and upper boundary respectively of the decision variables (Das and Dennis, 1998). These boundaries mostly represent safety precautions. For example: if the temperature T of a chemical reactor is a variable of the MOOP, this variable will be subject to boundaries so that the reactor does not overheat.

2.3 Solving methods: scalarisation and vectorisation

A MOOP has usually not one optimal solution but an infinite set of optimal solutions known as the Pareto front or the Pareto set. Finding or describing this Pareto front is the main goal when solving a MOOP. The human decision maker (DM) will choose one optimal solution from the Pareto front as a working point. It is desirable to produce a diverse Pareto set in a minimal computing time (Vallerio et al., 2015). Two major classes of MOOP solving methods exist: scalarisation methods and vectorisation methods (Logist et al., 2010).

Scalarisation methods employ deterministic algorithms and translate the MOOP into a set of single-objective optimisation problems (SOOP) via the use of weight factors. The SOOP are then minimised independently from the other objectives. This renders one non-dominated solutions for each SOOP. Solving the set of SOOP eventually results in the calculation of several solutions that are located on the Pareto front (Logist et al., 2010).

Deterministic algorithms solve a certain problem via a sequence of solutions which are known upfront and have a causal connection, i.e. the current generated solution results from a previously generated solution. The advantage of this method is that these algorithms can handle large numbers of decision variables and constraints. The major drawback of deterministic algorithms is that they incline to converge to local optima. Other drawbacks of some deterministic algorithms is that they cannot reach non-convex areas of the Pareto front or that these algorithms generate non-Pareto solutions (i.e. solutions that are not located

on the Pareto front but are dominated by others) (Logist et al., 2010).

Vectorisation methods use stochastic algorithms to solve a MOOP and tackle the MOOP as a whole. Vectorisation methods can be easily implemented and do not tend to converge to local optima, but are mostly unable to deal with complex constraints. Because all the objectives must be repeatedly compared, the algorithm can be time consuming when handling many-objective problems (i.e. multi-objective problems with four or more objectives). Thus vectorisation methods are practically limited to low dimensional search spaces (Hashem et al., 2017; Logist et al., 2010).

Although this thesis focuses on the use of a specific subset of the stochastic algorithms to solve multi-objective optimisation problems, scalarisation solving methods are widely used as well. Their major advantages are that they can be easily run computer wise, accompanied with relatively short computing time, and are therefore suitable to solve a MOOP. The three most commonly used scalarisation methods are briefly presented in the next sections. The concerned methods are the *Weighted sum* method, the *Normal boundary intersection* method and the *(enhanced) Normalised normal constraint* method.

2.3.1 Weighted sum method

The weighted sum method (WS) is the most widely used scalarisation method to solve a MOOP because of its simplicity. The method is based on combining the multiple objectives into a single convex combination of the objectives, via a weight factor w_i for each objective function f_i. The multi-objective problem as described in Equation (2.2) is translated to minimising the convex combination of the objectives (Das and Dennis, 1997):

$$\min_{x \in C} \sum_{i=1}^{M} w_i f_i(x) = w^T F(x) \tag{2.7}$$

M weights w_i are chosen with $0 \leq w_i \leq 1$ and $\sum_{i=1}^{M} w_i = 1$. w^T is a $1 \times M$ matrix and $F(\mathbf{x})$ is a $M \times 1$ matrix. The solution x^* of Equation (2.7) is also a solution of the original MOOP (Das and Dennis, 1997). The WS method is applied for an even spread of weights w_i in order to obtain different points on the Pareto front. Figure 2.1 graphically displays the overall process of the WS method.

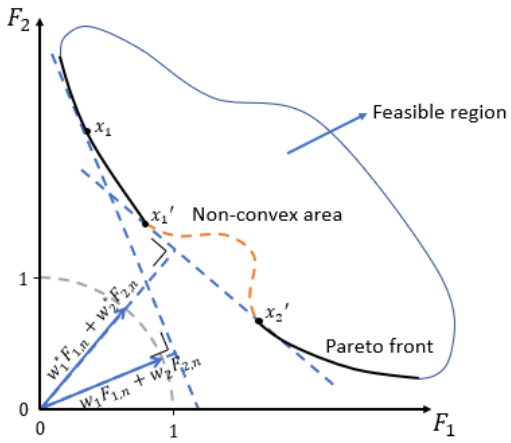


Figure 2.1: Graphical representation of the overall process of the WS scalarisation method for a bi-objective problem with objectives F_1 and F_2 (Das and Dennis, 1997).

Figure 2.1 displays a bi-objective problem for the sake of clarity. The Pareto front has a non-convex area which will prove to be a difficulty for the weighted sum method.

The objectives F_1 and F_2 as displayed in Figure 2.1 are normalised, rendering $F_{1,n}$ and $F_{2,n}$ respectively. The convex combination of the two normalised objectives can be seen as a vector of the unit circle with length 1 and angle $\arccos(1/w_1)$. w_1 is the weight corresponding to $F_{1,n}$ and w_2 is the weight corresponding to $F_{2,n}$. The tangent on the unit circle perpendicular to the constructed vector, is moved towards the feasible space, rendering the Pareto-optimal solution x_1. During each iteration the weights are altered, rendering only one Pareto-optimal solutions each time.

As already mentioned, reaching the Pareto-optimal solutions of the non-convex area of the Pareto front via the weighted sum method proves to be impossible. This is displayed in Figure 2.1 with the additional vector with a length of 1 and an angle of $\arccos(1/w_1')$. The tangent perpendicular to this vector renders the Pareto-optimal solutions x_1' and x_2'. The Pareto-optimal solutions of the non-convex area are never reached because no tangent can be constructed that is simultaneously perpendicular to a vector that is constructed via a convex combination of the considered objectives. To summarise, the two major shortcomings of the WS method are (Hashem et al., 2017):

1. Non-convex points of the Pareto front cannot be reached via this method.
2. An even spread of the weights w_i does not necessarily produce an even spread of points on the Pareto front.

New scalarisation methods were developed to circumvent these two shortcomings.

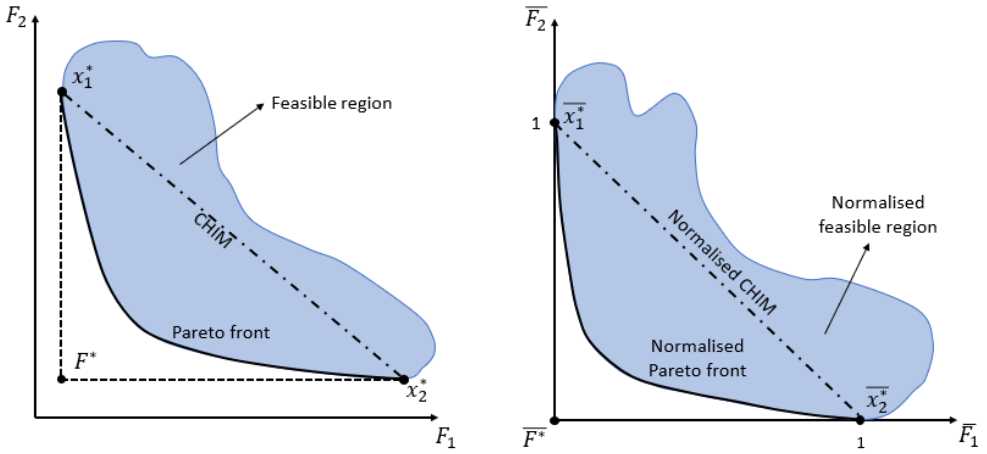
2.3.2 Normal boundary intersection method

The normal boundary intersection method (NBI) was developed to bypass the shortcomings of the WS method. In order to briefly describe the method, a few typical terms need explanation first.

The *Utopia point* F^* is defined as the vector in $\mathbb{R}^M$ that contains all the individual global minima f_i^* of the different objectives (Das and Dennis, 1998; Hashem et al., 2017):

$$F^* = \begin{bmatrix} f_1^* \\ \vdots \\ f_M^* \end{bmatrix} \tag{2.8}$$

Usually the Utopia point cannot be achieved by one single $\mathbf{x}$. Let x_i^* be the individual minimiser of f_i with $i = 1 \dots n$ then the *Convex Hull of Individual Minima* (CHIM) is defined as the hyperplane that connects all the individual minimisers x_i^* (Das and Dennis, 1998). The initial step of the NBI method consists out of the rescaling of all the individual objectives of the original MOOP so that the Utopia point is shifted towards the origin of the vector space $\mathbb{R}^M$ and all the individual minimisers of the different objectives are 1 unit away from the Utopia point (see Figure 2.2) (Messac et al., 2003).



(a) The initial design metric space **(b)** The rescaled design metric space

Figure 2.2: Rescaling of a bi-objective problem with objectives F_1 and F_2 (Messac et al., 2003).

Figure 2.2 displays a bi-objective optimisation problem for the sake of clarity. The individual minimisers of both objectives are respectively represented as x_1^* and x_2^* and the rescaled minimisers as $\overline{x_1^*}$ and $\overline{x_2^*}$ respectively. The Pareto front is situated on the edge of the feasible region that is closest to the Utopia point F^*. The feasible region is another term for the solution space C as defined in Equation (2.3).

The NBI-method continues by dividing the rescaled or normalised CHIM into several segments via the construction of quasi-normal lines on the CHIM (not shown in Figure 2.2). When the distance between the Utopia point and the CHIM is maximised along these quasi-normal lines, a Pareto-optimal point is found. It is clear that a uniform distribution of quasi-normal lines will lead to a uniform distribution of points on the Pareto front (Das and Dennis, 1998; Hashem et al., 2017). With this method it is also possible to reach non-convex points on the Pareto front. The major shortcomings of the NBI method however are (Das and Dennis, 1998):

1. Impossible to variate the resolution of the Pareto front.
2. Non-Pareto-optimal points can be produced.

Considering the first remark, it would be desirable to be able to alternate the resolution of the Pareto front, or at least of certain areas of the Pareto front. Only points and/or areas of the Pareto front that surpass an established trade-off value, are of interesting value to the DM. Generally these solutions are located on the steep segments of the Pareto front (Hashem et al., 2017). This problem will be tackled in the context of genetic algorithms during the course of this thesis via the development of novel genetic algorithms that take the trade-off of a solution into account.

2.3.3 (enhanced) Normalised normal constraint method

The (enhanced) normalised constraint method ((E)NNC) for solving a MOOP is in many aspects similar to the NBI-method. The Utopia point and individual minimisers of all the M objectives are determined. The feasible space and the M objectives are rescaled in the same way as described in 2.3.2. The rescaled individual minimisers are connected via a hyperplane in $\mathbb{R}^M$, the so-called *Utopia line* (the Utopia line is in every aspect the analogue of the CHIM). The major distinction between the NBI-method and (E)NNC-method is that the latter minimises one objective function while the other $M-1$ objective functions are incorporated as inequality constraints and thus reduce the feasible space. As a result of this reduction, many solutions are excluded, which increases the speed of the algorithm (Messac et al., 2003; Hashem et al., 2017).

2.4 Conclusion

In this chapter, the concept of multi-objective optimisation problems has been practically as well as mathematically introduced. The major incentive to optimise ((bio-)chemical) processes is to save money (be it in the form of energy, a higher yield, etc.). Multi-objective optimisation problems do not render a single optimal point but an infinite set of optimal points. Two major categories of solving methods have been developed to solve multi-objective optimisation problems and to generate Pareto-optimal solutions: scalarisation methods and vectorisation methods.

Scalarisation methods use deterministic algorithms to solve a MOOP and translate it into a SOOP which is then solved independently. This renders one Pareto-optimal solution. Via the use of weight factors, multiple SOOPs are formed in order to obtain a spread in solutions on the Pareto front. Vectorisation methods on the other hand, use stochastic algorithms and tackle the MOOP as a whole. A sub-field of stochastic algorithms are the evolutionary algorithms, which are thoroughly examined in the next chapter.

Chapter 3

Evolutionary Algorithms

3.1 Introduction

Multi-objective optimisation problems have been introduced both practically and mathematically in the previous chapter. Three commonly used scalarisation methods have been introduced simultaneously. However, throughout this thesis stochastic algorithms will be used to solve multi-objective optimisation problems. A sub-field of the stochastic algorithms are the evolutionary algorithms (EA), on which the overall focus of this thesis will be. In this chapter, the global philosophy and method of evolutionary algorithms is revised. Subsequently two specific evolutionary algorithms are explained in detail: NSGA-II and NSGA-III.

The use of evolutionary algorithms to solve multi-objective optimisation problems has proven its value over the last decades. Their general advantage over other solving methods is that these algorithms are able to generate multiple Pareto-optimal solutions in a single simulation run. Nonetheless early evolutionary algorithms received some criticism which led to the development of NSGA-II and eventually NSGA-III. The main critiques on the early evolutionary algorithms were their high computational complexity, lack of elitism and need to specify a sharing parameter (Deb et al., 2002).

3.2 General considerations

An evolutionary algorithm is a population-based optimisation method which draws its basic philosophy from biological evolution. Mechanisms such as reproduction, selection of the fittest, mutation, etc. that are commonly accepted in the context of biological reproduction and evolution, are also found in an evolutionary algorithm (Back, 1996).

In order to create a set of Pareto-optimal solutions of a multi-objective optimisation problem, a first (random) set of parent solutions is created. All the solutions p of that set are compared to one another and are sorted into several non-dominated fronts (NDF). The first non-dominated front contains all the solutions that do not dominate each other but dominate all the other solutions. The solutions of the second non-dominated front are dominated by the solutions of the first front, do not dominate each other, but dominate the solutions of the third front and so on.

Of this first generation of solutions, only the fittest solutions are retained. The fittest solutions are those that are closest to the Pareto front i.e. the solutions located in the first few non-dominated fronts. These solutions are combined with each other and are mutated in order to create a new generation of solutions. At their turn, these solutions are sorted into several non-dominated fronts after which the process repeats itself. Because after each iteration only the best solutions of each generation are retained, the set of

solutions will eventually converge to the Pareto front. When after several iterations the difference between the solutions of two subsequent generations becomes negligible, it can be concluded that the Pareto front has been reached. Figure 3.1 graphically displays the global philosophy of evolutionary algorithms.

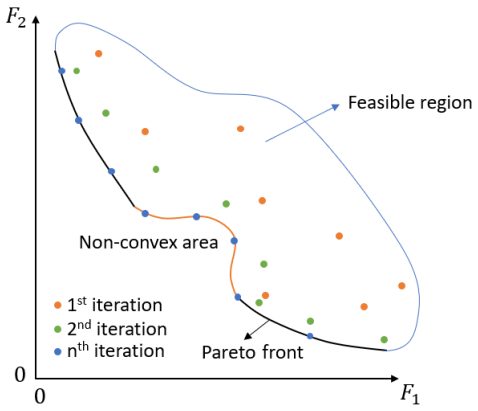


Figure 3.1: General representation of a solving method for multi-objective optimisation problems based on an evolutionary algorithm.

The solutions displayed in Figure 3.1 as the solutions of the first iteration, are the randomly generated solutions within the feasible region. Via subsequent selection, mutation and crossover actions during the following iterations, the generated solutions move closer to the Pareto front. The solutions of the final n-th iteration are located in the close proximity of the Pareto front. Non-convex areas of the Pareto front can be reached as well as convex areas and a uniform spread of solutions can be maintained.

The high computational complexity of the early algorithms was a consequence of the used non-dominated sorting approach. Computational complexities reached as high as $O(MN^3)$ (with M the number of objectives and N the size of the solution-population). These computational complexities were reduced in newer algorithms via the use of less complex sorting approaches. Other improvements seen in these algorithms, is their use of elitism and crowded comparison. Elitism entails that the best individuals of a generation are kept unchanged in the next generation. This leads to a faster convergence to the Pareto front. The sharing parameter of the early algorithms was replaced by the crowded comparison approach to keep enough diversity amongst the solutions in a set (Deb et al., 2002).

Thanks to these three major improvements, new algorithms like NSGA-II and NSGA-III have grown very popular and are commonly used. Nonetheless further improvements can still be made. For instance, it would be desirable to create a Pareto front with an adaptable solution resolution. The Pareto front can namely be subdivided in areas containing high trade-off solutions and areas containing low trade-off solutions. High trade-off solutions are of bigger interest for the decision maker and are located on steep segments of the Pareto front while solutions with a low trade-off are located on the flat sections (Hashem et al., 2017). If the decision maker decides to switch between one Pareto-optimal solution and another Pareto-optimal solution, a trade-off occurs. Because of the switch, the value of several objectives will increase (i.e. become less optimal) while those of other (targeted) objectives will decrease. A switch to a solution with a high trade-off is characterised by a large decrease in values of one or more objectives while the increase in value of other objectives is negligible. Because of this high trade-off, and although several qualities will have been lost, the switch is profitable.

The following chapters will delve deeper into the trade-off of generated solutions and how to eventually obtain a Pareto front with an adaptable resolution. Firstly, the NSGA-II and NSGA-III algorithms, as mentioned before, are thoroughly discussed.

3.3 Non-dominated sorting genetic algorithm II

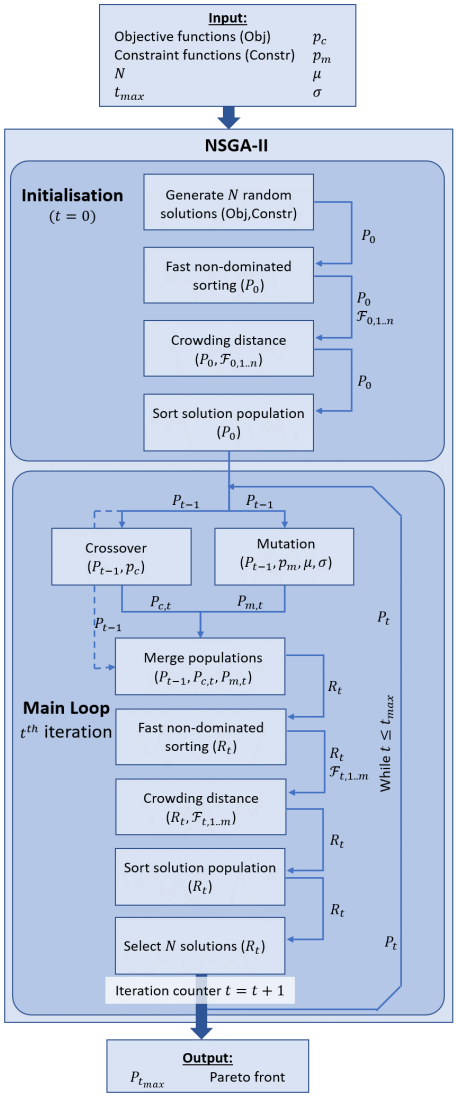


Figure 3.2: Flow sheet of the NSGA-II algorithm.

The flow sheet of the NSGA-II algorithm is represented in Figure 3.2. NSGA-II is a multi-objective evolutionary algorithm developed as an answer to the shortcomings of the early EA's. It uses non-dominated sorting and sharing, like all other EA's, but it was one of the first EA's to employ elitism and crowded comparison. NSGA-II uses a novel fast non-dominated sorting mechanism which allowed the reduction of the computational complexity of the algorithm to $O(MN^2)$ (Deb et al., 2002).

3.3.1 The main loop

During the first iteration, N initial random solutions are generated and are put into the parent set P_0. The solutions in this parent set are sorted according to their fitness via a fast non-dominated sorting algorithm and a crowding distance calculation. Hereupon the sorted solutions are used to form an offspring set Q_1 ($Q_1 = P_{c,1} \cup P_{m,1}$ and $|Q_1| = N$). Offspring solutions are achieved via selections, recombinations and mu-

tations of the solutions in the parent set. The non-dominated sorting of the parent set is only a separate step during the first iteration. The solutions of the parent sets of all the following iterations have already been appointed their corresponding non-dominated rank during the previous iteration (Deb et al., 2002).

Elitism implies that the best solutions of the parent set of the previous iteration are kept unchanged during the current iteration. NSGA-II obtains this by merging both the parent set P_{t-1} and the corresponding offspring set Q_t of the t^{th} iteration into a combined population R_t and $|R_t| = 2N$ (Deb et al., 2002). All the $2N$ solutions of the combined population are sorted in their corresponding non-dominated fronts and their crowding distance is calculated. Then, based on the rank of the non-dominated front to which they belong and their crowding distance, the best N solutions are retained. These solutions form the P_t parent set, i.e. the parent set of the t-th iteration. Note that when the non-dominated fronts together contain more than N solutions, further non-dominated sorting is futile because these extra solutions will be rejected in the selection step (Deb et al., 2002).

To solve a MOOP via NSGA-II the number of iterations, the size of the solution population and the conventional EA-parameter like the crossover probability and mutation probability must be pre-defined by the user.

3.3.2 Recombination or crossover

The solution population set P_t of the t-th iteration is used as the parent solution population of the $(t+1)$-th iteration. One of the possibilities to create new solutions is via the recombination, or crossover, of two randomly picked solutions. Two parent solutions form two offspring solutions. The crossover action can be compared to biological meiosis. The user pre-defines the probability p_c of a crossover to occur. This probability p_c is one of the so-called conventional EA-parameters that have to be pre-defined by the user before the algorithm can be initialised (Valadi and Siarry, 2014). The optimal value of p_c is problem dependent but both Valadi and Siarry (2014) and Deb et al. (2002) propose a crossover probability p_c of 90.00 %.

Two random parent solutions are selected from the parent solution set with a probability of p_c. The values of the process variables of these two randomly selected solutions are used to generate the values of the process variables of the two offspring solutions. The crossover algorithm as used in NSGA-II and NSGA-III is represented in Algorithm 1 (Valadi and Siarry, 2014):

Algorithm 1 Crossover (Kalami, 2015, 2016)

Require: p, q
$N = \texttt{size}(p)$
$\alpha = \texttt{rand}(N, 0, 1)$
$p' = \alpha \times p + (1-\alpha) \times q$
$q' = \alpha \times q + (1-\alpha) \times p$
return p', q'

α is a randomly generated $1 \times N$ matrix operator containing random numbers between 0 and 1. The number of variables of α equals the number of process variables of p (and q). The offspring solutions p' and q' created via the crossover algorithm are linear recombinations of the two parent solutions p and q. This process is graphically displayed for offspring solution p' in Figure 3.3 for a bi-objective problem for the sake of clarity. The corresponding α-operators are denoted α_1 and α_2 for the F_1- and F_2-variables respectively.

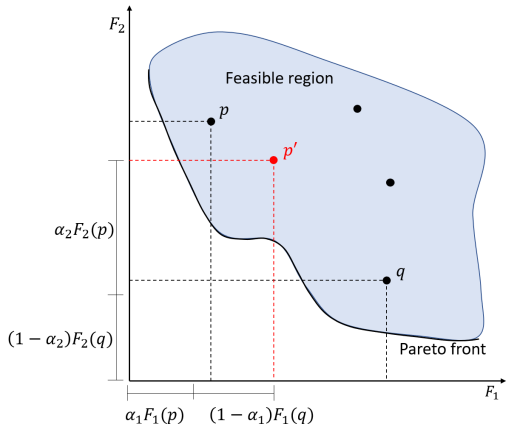


Figure 3.3: Graphical representation of the crossover process for a bi-objective problem with objectives F_1 and F_2.

The crossover algorithm generates two solutions that are in the proximity of the two parent solutions. This allows the algorithm to further explore already discovered areas of the Pareto front. Moreover, the two parent solutions do not necessarily have to be two random selected solutions. If it is opted to only crossover solutions that are, for instance, already in each others proximity or that are part of the same non-dominated front (see 3.3.4), the generated solutions will have more specific properties. Implementing such additional restriction allows the user to direct the algorithm to certain areas of the Pareto front or increase the convergence speed (Valadi and Siarry, 2014).

Especially a high convergence speed is a desirable algorithm property. The convergence speed of the algorithm will decrease if the generated offspring solutions are located further away from the Pareto front than their parents. By randomly selecting two parent solutions from a parent solutions set containing only members who are already in each others proximity, this risk is reduced (Valadi and Siarry, 2014).

Although the crossover algorithm is very suitable to generate new offspring solutions, it is not designated to preserve sufficient diversity amongst the generated solutions. As the number of iterations increases, the solutions will become increasingly more related to each other if only a crossover functions is used to generate new solutions. To obtain and preserve sufficient diversity amongst the solutions, a second solution generating function is used: the mutation function (Valadi and Siarry, 2014).

3.3.3 Mutation

In biological genetics it is possible that during mitosis or meiosis a mutation of one or multiple genes occurs. This eventually gives rise to the development of new species and subspecies. The mutation algorithm used in evolutionary algorithms is used for exact same reason. It allows the algorithm to diverse its solution population set and generate new offspring solutions with very different properties in comparison to their parent solution (Valadi and Siarry, 2014). Algorithm 2 represents the mutation algorithm as used in NSGA-II and NSGA-III.

Algorithm 2 Mutation (Kalami, 2015, 2016)

Require: p, μ, σ
  $N = \texttt{size}(p)$
  $nVar = \lceil \mu \times N \rceil$
  $\Delta Var = \texttt{randsample}(N, nVar)$
  $p' = p$
  for all members of ΔVar **do**
    $p'_i = p'_i - \sigma \times \texttt{rand}(0,1)$
  end for
  return p'

The $\texttt{randsample}(N, nVar)$ function returns a $nVar \times 1$ matrix containing random selected integers, without replacement, between 1 and N. $\texttt{rand}(0,1)$ returns a random value between 0 and 1, denoted as β_i. p'_i stands for the i-th variable of solution p'. A random solution p is selected from the parent solution population set with a probability of p_m. Deb et al. (2002) and Liagkouras and Metaxiotis (2017) propose a mutation probability p_m of $1/n$, with n the number of process variables. On a more general note, the mutation probability p_m is, just like the crossover probability p_c, dependent on the MOOP in question. The mutation process is graphically represented in Figure 3.4 for a bi-objective problem for the sake of clarity. In the represented process only one variable is mutated, $nVar = 1$.

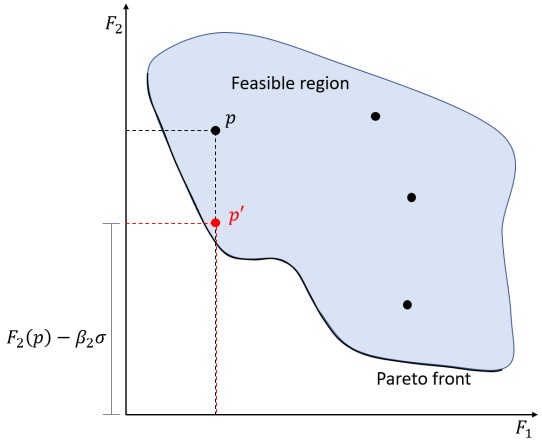


Figure 3.4: Graphical representation of the mutation process for a bi-objective problem with objectives F_1 and F_2.

Together with p_m, μ and σ have to be pre-defined by the user. μ is the mutation rate and specifies how many variables are mutated of a selected parent solution p. σ is the mutation step size and defines the magnitude of the mutations. If p_m, μ, and σ all have large values, a lot of diverse solutions will be generated, maintaining a large diversity in the solution population. However, it can be expected that the convergence speed in this scenario will be low (Valadi and Siarry, 2014).

Eventually, the offspring solutions of the t-th iteration, generated via mutation and crossover, are bundled in the offspring set Q_t. NSGA-II and NSGA-III are elitist algorithms thus the parent solution set P_{t-1} is not discarded. By doing so it is possible to retain the best solutions of the previous iteration unchanged in the current one, and simultaneously increasing the convergence speed of the algorithm. The solutions of the combined solution set R_t have to be sorted according their fitness in order to select the fittest ones. NSGA-II and NSGA-III use a fast non-dominated sorting algorithm theretofore (Deb et al., 2002; Deb and Jain, 2014).

3.3.4 Fast non-dominated sorting

As already mentioned, the early EA's had often a computational complexity as high as $O(MN^3)$ because they all used a rudimentary non-dominated sorting approach. In order to sort all the (randomly) generated solutions into their corresponding non-dominated fronts, each solution had to be compared to all the others and this occurred for each objective. This led to a computational complexity of $O(MN)$ in order to determine if a solution is non-dominating or not. Doing this for each of the N solutions led to a complexity of $O(MN^2)$. At this point only the first non-dominated front had been determined. In the worst case each of the N solutions is part of a different non-dominated front, so to determine all the non-dominating fronts, MN^3 comparison were needed or, in other words, the computational complexity of the complete sorting algorithm could reach as high as $O(MN^3)$ (Deb et al., 2002).

NSGA-II uses a fast non-dominated sorting algorithm which allows its computational complexity to be lowered to $O(MN^2)$ in the worst case, i.e. all the N solutions are in different non-dominated fronts. The initial steps of the fast non-dominated sorting algorithm are identical to that of the rudimentary non-dominated sorting algorithm as described above. The objective values of each solution i are compared to those of all the other solutions in the solution population. This requires MN^2 comparisons. Whilst doing this, for each solution i it is calculated by how many solutions it is dominated. This leads to the domination counter n_i. All the solutions that are dominated by i are put into a separate set S_i (Deb et al., 2002).

Considering the t-th iteration, all the solutions with a domination counter $n_i = 0$ are non-dominated and are members of the first non-dominated front $\mathcal{F}_{t,1}$. The domination counters of all the members of their sets S_i are reduced by one. If, by doing so, the domination counter of one or more of these solutions becomes zero, then these solutions are members of the second non-dominated front $\mathcal{F}_{t,2}$. Each time members of a non-dominated front are determined, they are put in a separate set, leading in a steadily decrease in size of the original solution space. When a solution is appointed to a non-dominated front, it is never visited again (Deb et al., 2002). In the worst case the domination counter of a solution adds up to $N-1$ which means that this solution is, after being visited N times during the initial comparisons, revisited $N-1$ times before its domination counter equals zero and that it can be appointed to its non-dominated front. In total N^2 visits and MN^2 comparisons are needed to determine all the non-dominated fronts so the computational complexity is $O(MN^2)$ in a worst case scenario (Deb et al., 2002).

3.3.5 Crowded comparison

The solutions of a MOOP obtained via some scalarisation methods tend to converge to local optima. In order to prevent this from happening when an EA is used, and to maintain a good spread in solutions, solution density management is needed. In early EA's this was obtained by using a sharing parameter. This parameter represents the distance two solutions should be apart in order to be considered two different solutions. Solutions that were closer to each other than the sharing parameter were removed from the solution population. The efficiency of this work method was highly dependent on the value of the chosen sharing parameter. Another major drawback of this approach was that the distance between all the solutions had to be calculated, resulting in an overall computational complexity of $O(N^2)$ (Deb et al., 2002).

NSGA-II uses a crowded comparison method instead of sharing parameter. This abolishes the necessity of a user-defined (sharing) parameter. The new crowded comparison method contains two major parts:

1. Estimating the density of the solutions
2. Selecting solutions to obtain a satisfactory spread

Firstly in order to obtain an estimate of the density of solutions around a solution i, the average distance between two solutions on either side of the solution i is calculated in the direction of each objective. This average distance is defined as the crowding distance i_d and is the average length of the sides of the cuboid formed by using the closest neighbouring solutions as vertices (Deb et al., 2002). This is represented in Figure 3.5. In this example, for the sake of clarity, only two objectives are used and thus $i_d = \frac{d_1+d_2}{2}$.

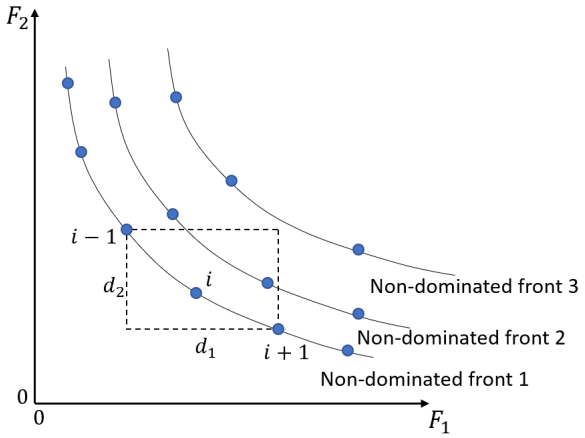


Figure 3.5: Estimating the density of solutions around a solution i for a bi-objective problem with objectives F_1 and F_2 (Deb et al., 2002).

To be able to determine the neighbouring solutions in the direction of an objective, all the solutions of the same non-dominated front must be sorted in ascending order in regard to the objective direction and all the objectives must be normalised. When the obtained crowding distance of a solution is small, it can be concluded that this solution is more crowded by other solutions. In the following step, the more crowded solutions will be sorted out of the solution space to obtain an even spread in Pareto-optimal solutions (Deb et al., 2002).

3.3.6 Solution selection procedure

At this point in the main loop, each solution is now sorted into a non-dominated front and is appointed its crowding distance. To determine which solutions to retain in the eventual Pareto-optimal solutions set, a comparison is needed between these two properties. Only a number of solutions are selected to be retained. The offspring set P_t generated during the t-th iteration of the main loop must have the same size as the initial parent set P_{t-1}, i.e. N solutions. Until the size of the offspring set $P_t = N$, solutions are added in the following order (Deb et al., 2002):

1. Solutions from a lower non-dominated front are preferred
2. Solutions with a bigger crowding distance are preferred

Firstly all the members of the first (or lowest) non-dominated fronts are appointed to the solution set P_t. If however adding a complete extra non-dominated front would surpass N, then only the solutions with the highest crowding distance of this particular non-dominated front are added to the solution set P_t until it contains N solutions. To simplify this procedure, the solutions in the combined solution set R_t are sorted based on their non-dominated rank and crowding distance. R_t can be considered as an array containing $2N$ members, each with multiple properties. During the initial step of the solution selection procedure, the solutions of R_t are sorted in descending order according to their crowding distance. The solutions of the achieved array are subsequently sorted in ascending order according to their non-dominated rank. The

sorted R_t set (or array) now consists out of solutions with low non-dominated ranks and high crowding distances in the first array positions. The selection of N solutions with the lowest non-dominated rank and highest crowding distance is now reduced to selecting the first N solutions of the sorted combined solution set R_t. This process is graphically displayed in Figure 3.6. By way of illustration, a simplified combined solution set R_t is provided. Note that the non-dominated rank and crowding distance of a solution should be seen as inherent properties of the solution itself. This is also applies to the value of the process variables and the corresponding objective functions values.

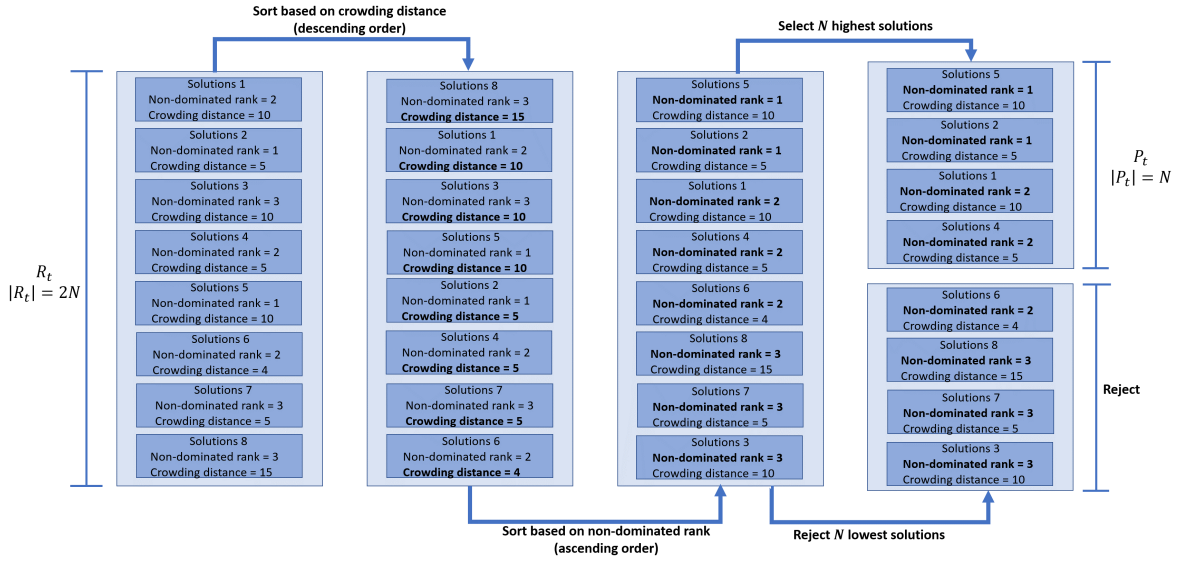


Figure 3.6: Graphical representation of the population sorting process based on non-dominated rank and crowding distance for a thought experiment (Deb et al., 2002).

In Figure 3.6, $N = 4$. Only the solutions in the top 4 array positions are selected to form the solution set P_t. The last N solutions of the sorted combined solution set R_t are rejected. Note that this might not be the best approach to select solutions because little attention is given to the crowding distance of the selected solutions. Even if all the solutions of the first $k-1$ non-dominated fronts have small crowding distances, they will be added to the solution set P_t nonetheless because of their low non-dominated rank. Only the solutions of the last added non-dominated front $\mathcal{F}_{t,k}$ are selected based on their crowding distance.

3.3.7 Constraints

In the context of using NSGA-II (and -III) to optimise real ((bio-)chemical) processes via multi-objective optimisation, it is important to be able to verify the obtained solutions to certain constraints. In the discussion so far, no attention has been given to working with a constrained feasible solution space. If constraints are taken into account, the definition of domination has to be altered. A solution p constraint-dominates a solution q if one of the following is true (Deb et al., 2002):

1. Solution p is feasible and solution q is infeasible
2. Solution p and q are both feasible and solution p dominates solution q (p is member of a lower non-dominated front or has a smaller crowding distance than solution q)
3. Solution p en q are both infeasible but solution p violates the constraints less than solution q

By adding a so-called feasibility counter $FC(\mathbf{x})$ to the fast non-dominated sorting algorithm, the feasibility of solutions in regard to constraints can be taken into account. If a solution is feasible, the feasibility counter is set to zero. If a solutions breaches a constraint, the feasibility counter $FC(\mathbf{x})$ equals the summation of all the normalised constraint breaches. Normalising the constraint functions $g(\mathbf{x}) \leq c_1$ and $h(\mathbf{x}) = c_2$ is done by dividing the constraint functions by the constant term (see Equation (3.1) and (3.2) respectively):

$$g_n(\mathbf{x}) = \frac{g(\mathbf{x})}{c_1} - 1 \leq 0 \tag{3.1}$$

and

$$h_n(\mathbf{x}) = \frac{h(\mathbf{x})}{c_2} - 1 = 0 \tag{3.2}$$

A breach of the normalised inequality constraint function will, in this case, yield a positive value. If the normalised inequality constraint function yields a negative value, this value should be converted to zero because then there is no breach in terms of inequality constraints. This is obtained by the $\langle\alpha\rangle$ operator which returns α if $\alpha > 0$ and returns zero if $\alpha \leq 0$.

The normalised equality constraint function can yield in case of a breach both a positive or negative value. Only if the obtained value equals zero, there is no breach in terms of equality constraints. By only taking the absolute value of the obtained value into account, the feasibility counter will amount to a positive value in case of a breach and will equal zero in case the solution $\mathbf{x}$ is feasible and does not violate any constraint functions. Thus, the feasibility counter $FC(\mathbf{x})$ can be defined as (see Equation (3.3)) (Jain and Deb, 2014):

$$FC(\mathbf{x}) = \sum_{i=1}^{ni} \langle g_{n,i}(\mathbf{x}) \rangle + \sum_{j=1}^{ne} |h_{n,j}(\mathbf{x})| \tag{3.3}$$

This method for handling constraints can be used for any EA (Deb and Jain, 2014; Jain and Deb, 2014).

3.4 Non-dominated sorting genetic algorithm III

Non-dominated sorting genetic algorithm III (NSGA-III) was developed in order to handle many-objective problems more easily ($M \geq 4$). Its basic structure is similar to that of NSGA-II but it uses a different approach to keep diversity in the solution set and uses a different solution selection procedure. The made alternations ensue from several difficulties that arise when handling many-objective problems. The major difficulties that need consideration are listed below (Deb and Jain, 2014):

1. With increasing objectives, the amount of generated solutions that are non-dominated increase. Because NSGA-II uses an elitist approach, the space for generating new solutions becomes smaller because the best solutions are retained unchanged in the next generation.

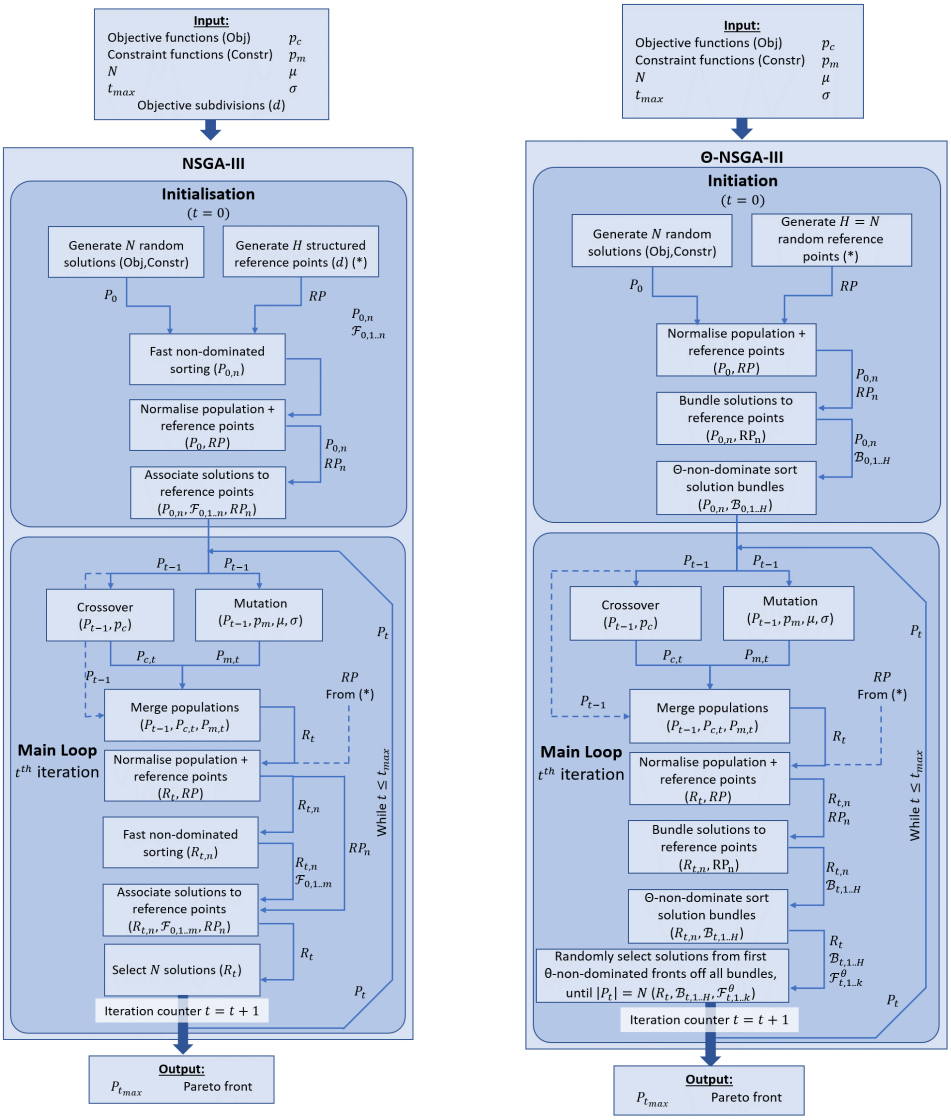
2. The calculation of the crowding distance becomes increasingly more difficult and computational more expensive when working in a large dimensional space.

3. It is not uncommon when working with many-objective problems that solutions of the parent set P_{t-1} are isolated from each other. The recombination of such solutions produces offspring solutions that are, in their turn, isolated from their parents.

NSGA-III alleviates these difficulties via the use of reference points that are uniformly spread out in the search space. However, NSGA-III still uses non-dominated sorting as its fundamental approach to select the best solutions of the t-th generation. Mainly because of the first difficulty as listed above, and even with the use of reference points, the obtained solutions after the predefined number of iterations are highly diverse but the convergence of NSGA-III to the Pareto front can be slow. To alleviate this disadvantage,

an alternative θ-non-dominated sorting genetic algorithm III (θ-NSGA-III) was developed. This algorithm uses θ-non-dominated sorting instead of non-dominated sorting. This allows θ-NSGA-III to have a better convergence to the Pareto front whilst retaining the high solution-diversity of NSGA-III (Yuan et al., 2014). Because of this significant improvement, both NSGA-III and θ-NSGA-III will be presented in the following sections.

3.4.1 The main loop



(a) NSGA-III **(b)** θ-NSGA-III

Figure 3.7: Flow sheet of the (a) NSGA-III algorithm (Deb and Jain, 2014) and (b) the θ-NSGA-III algorithm (Yuan et al., 2014).

NSGA-III and θ-NSGA-III, like NSGA-II, do not need extra input besides the conventional EA parameters like the solution population size, number of iterations, and the parameters concerning the recombination and mutation of parent solutions in order to generate offspring solutions. The flow sheets of the NSGA-III and θ-NSGA-III algorithms are represented in Figure 3.7(a) and 3.7(b) respectively.

The first steps in the NSGA-III and θ-NSGA-III main loops do not differ much from each other. The parent solutions set P_{t-1} ($|P_{t-1}| = N$) is used to form an offspring solution set Q_t ($Q_t = P_{c,t} \cup P_{m,t}$ and $|Q_t| = N$) and both sets are merged into the combined solution set $R_t = P_{t-1} \cup Q_t$ and $|R_t| = 2N$. In the NSGA-III algorithm, the combined solution set R_t is sorted into its various non-dominated fronts, via a fast non-dominated sorting algorithm, until the non-dominated fronts contain together more than N solutions. With regard to generating the parent set P_t for the $(t+1)$-th iteration, N solutions must be selected from the combined solution set R_t.

θ-NSGA-III does not use a non-dominated sorting algorithm but bundles all the $2N$ solutions in R_t around various reference points and their corresponding reference lines. The solutions in each bundle are then sorted according to their fitness using a θ-non-dominance sorting algorithm. This algorithm also stresses the convergence of the generated solutions to the Pareto front. Two solutions that are in the same non-dominated front in NSGA-III (and thus have the same fitness), can have a different θ-non-dominance. When a solution is closely located to the Pareto front, it will have a higher θ-non-dominance than a solution that is further away form the Pareto front. The convergence of a solution to the Pareto front is estimated via the distance between the perpendicular projection of the solution on its reference line and the origin of the objective space. NSGA-III only uses the distance between the solution itself and its perpendicular projection on its reference line as a replacement for the crowding distance of NSGA-II. No attention is given to the convergence of solutions to the Pareto front (Yuan et al., 2014; Deb and Jain, 2014).

The NSGA-III algorithm continues by sorting the selected solutions into a temporary parent solution set S_t until $|S_t| \geq N$. Solutions located in the lowest non-dominated fronts $\mathcal{F}_{t,i}$ are given priority. If the non-dominated front $\mathcal{F}_{t,k}$ is the last non-dominated front to be sorted into S_t and for which $|S_t| \geq N$ for the first time, then two situations are possible: (i) $|S_t| = N$ and then $P_t = S_t$ or (ii) $|S_t| > N$.

In the second situation the first $k-1$ non-dominated fronts are sorted into P_t. From the k^{th} non-dominated front only those solutions are selected that maintain the highest solution diversity. In NSGA-II this was achieved via the crowding distance, but in NSGA-III solutions are associated to reference points rather than calculating their crowding distance. This procedure is more thoroughly explained in section 3.4.4. If more than one solution is associated to a reference point, only the solution with the shortest perpendicular distance to the according reference line, is sorted into the parent set P_t. This process is conducted for all the reference points (and, if necessary, repeated) until $|P_t| = N$. This parent solution set is send to the next, $(t+1)$-th iteration and the main loop is repeated (Deb and Jain, 2014).

Because θ-NSGA-III uses a θ-non-dominance sorting algorithm (see 3.4.5) rather than a non-dominated-sorting algorithm, the selection of the fittest solutions can be reduced to selecting random solutions from the first θ-non-dominated fronts of each reference points until $|P_t| = N$. This random selection procedure is possible because the allocation of the $2N$ solutions of the combined solution set R_t to their corresponding reference point, already guaranteed solution diversity (Yuan et al., 2014). By implementing convergence as an extra selection parameter, the solutions generated via θ-NSGA-III will converge quicker to the Pareto front than the solutions of NSGA-III.

3.4.2 Reference points

Reference points are used to structure the search space. They can be generated via a systematic approach or they can be preferential points provided by the DM. When it is opted to generate reference points, Das and Dennis (1998) offers an approach in which H reference points are constructed. The reference points are situated on the hyperplane that intersects all the normalised objective vectors at a

value of one. This approach is used in NSGA-III. The amount of reference points that are generated is dependent on the number of objectives M (Das and Dennis, 1998):

$$H = \frac{(M+d-1)!}{d!(M-1)!} \tag{3.4}$$

In Equation (3.4), d stands for the amount of subdivisions made alongside each objective. The size N of the solution population is related to the amount of reference points H, as $N \approx H$. This is in aspiration that approximately one generated solution can be linked to one reference point. When working with many-objective problems however, which is the major target of NSGA-III, it is clear that H, and thus N, can become very large. This is a first complicated feature of the NSGA-III algorithm (Yuan et al., 2014). Figure 3.8 represents the structured reference points located on the normalised hyperplane of a three-objective problem and $d = 5$.

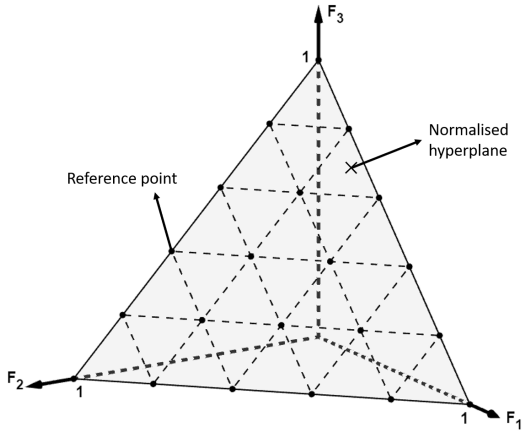


Figure 3.8: Graphical representation of the structured reference points of the NSGA-III algorithm of a three-objective problem with objectives F_1, F_2, and F_3 (Deb and Jain, 2014).

Considering the problem represented in Figure 3.8 is a three-objective problem ($M = 3$) and it is opted to divide each objective in 5 subdivisions ($d = 5$), Equation (3.4) states that the number of generated reference points $H = \frac{(3+5-1)!}{5!(3-1)!} = 21$. This is indeed confirmed via Figure 3.8.

θ-NSGA-III however uses a more flexible but comparable approach to construct H reference points. The solution population size N can now be chosen and the number of reference points $H = N$. The reference points h_i are randomly generated M-dimensional points defined as $h_i = (h_{i,1}, h_{i,2}, \ldots, h_{i,m})^T$ (Yuan et al., 2014).

In both algorithms, generated solutions are associated to the reference points via reference lines which connect reference points to the origin of the multidimensional space. The main objective of the reference points is to assist in the maintenance of diversity in the solution population. They substitute the crowding distance used in NSGA-II.

In NSGA-III, when selecting the solutions for the parent set P_t of the next iteration, priority is given to non-dominated solutions and, secondarily, to solutions that are in the proximity of a reference point. The θ-NSGA-III algorithm allocates all the solutions of the combined set R_t to their corresponding reference points. In both cases, the allocation of a solution to a reference point is conducted in a similar way and this is explained in section 3.4.4 (Deb and Jain, 2014).

3.4.3 Normalisation of solutions

Because the reference points of NSGA-III are located on a normalised hyperplane, the solutions in the temporary parent solution set S_t must be normalised first before they can be correctly allocated to a reference point. The second complicated feature of the NSGA-III algorithm is however that this *Adaptive normalisation method* as presented by Deb and Jain (2014) is a very elaborate and a computational highly demanding method. Yuan et al. (2014) proposes to use a more intuitive normalisation method which has better results than the elaborate normalisation method of Deb and Jain (2014) when handling many objectives. NSGA-III will only normalise the solutions of S_t, while θ-NSGA-III normalises all the solutions in R_t.

Both normalisation methods of NSGA-III and θ-NSGA-III start with the the determination of the ideal point $\bar{a}$ of the set S_t or R_t respectively. $\bar{a}$ is defined as (Deb and Jain, 2014; Yuan et al., 2014):

$$\bar{a} = (a_1^{min}, a_2^{min}, \ldots, a_M^{min})^T \tag{3.5}$$

It is possible to calculate the exact value of $\bar{a}$ but this would be a very computational demanding step. In Equation (3.5), a_i^{min} (and $i = 1, \ldots, M$) stand for the minimum value of the i-th objective function that has been generated thus far. This value is updated when a new minimum value is found. Along with the ideal point, the maximum values a_i^{max} (and $i = 1, \ldots, M$) of all the M objective function f_i are determined from the solution set S_t (or R_t in the case in the case of θ-NSGA-III) and $a^{max} = (a_1^{max}, \ldots, a_M^{max})^T$ (Deb and Jain, 2014; Yuan et al., 2014).

The normalisation method of NSGA-III continues by first subtracting each objective f_i by its corresponding minimum value a_i^{min}: $\bar{f}_i = f_i - a_i^{min}$. By doing so, the ideal point $\bar{a}$ is translated to the origin of the M-dimensional search space. Then, an M-dimensional hyperplane is constructed containing all the translated maximum values $\bar{a}_i^{max}$. The intercepts of this M-dimensional plane with the translated objectives $\bar{f}_i$ are defined as a_i. The eventual normalised objective f_i' is defined as (Deb and Jain, 2014):

$$f_i'(\mathbf{x}) = \frac{f_i(\mathbf{x}) - a_i^{min}}{a_i} = \frac{\bar{f}_i(\mathbf{x})}{a_i} \tag{3.6}$$

The adaptive normalisation method as used in the NSGA-III algorithm is represented in Figure 3.9.

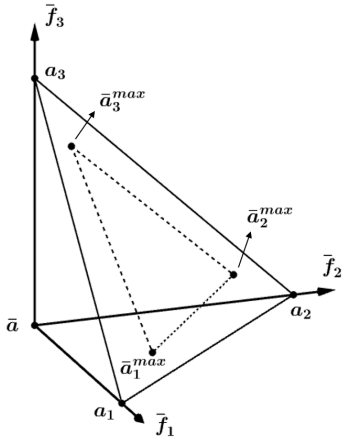


Figure 3.9: Graphical representation of the adaptive normalisation method as used in the NSGA-III algorithm, for a three-objective problem with objectives f_1, f_2, and f_3 (Deb and Jain, 2014).

Figure 3.9 represents the second step of the adaptive normalisation method. At this point, the objective functions f_i and the maximum values a_i^{max} have already been translated.

θ-NSGA-III uses a more intuitive normalisation method (Yuan et al., 2014):

$$f_i'(\mathbf{x}) = \frac{f_i(\mathbf{x}) - a_i^{min}}{a_i^{max} - a_i^{min}} \tag{3.7}$$

In this normalisation method, as presented in Equation (3.7), there is no need to calculate the intercept points of an M-dimensional hyperplane, which severely diminishes the computational complexity of the proposed normalisation algorithm.

3.4.4 Bundling of solutions around reference points

For each normalised solution, the metric perpendicular distances d_i between the solution and the different reference lines is calculated. The solution is allocated to the reference point for which its metric perpendicular distance to the corresponding reference line is the smallest. This allocation process is the same in both NSGA-III and θ-NSGA-III but the sets of which the solutions are allocated, are different (S_t and R_t respectively). Figure 3.10 graphically displays the bundling process.

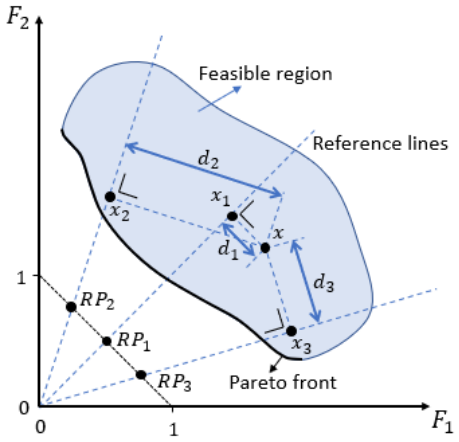


Figure 3.10: Graphical display of the bundling process of NSGA-III and θ-NSGA-III (Yuan et al., 2014).

In Figure 3.10, the distance d_1 is the smallest metric distance so the concerned solution x will be allocated to reference point RP_1. If H reference points are created, H cluster sets $\mathcal{B}_i$ of solutions are generated after the bundling process, with $\mathcal{B}_i$ the cluster set corresponding to the i-th reference point. $|\mathcal{B}_{1..H}| = |S_t|$ in the case of NSGA-III and $|\mathcal{B}_{1..H}| = |R_t|$ in the case of θ-NSGA-III (Yuan et al., 2014; Deb and Jain, 2014).

3.4.5 Non-dominated sorting or θ-non-dominated sorting

While NSGA-III uses the same fast non-dominated sorting algorithm as NSGA-II (see 3.3.4), θ-NSGA-III uses θ-non-dominated sorting.

When two solutions x_1 and x_2 of the combined solution set R_t are considered, then x_1 θ-dominates x_2 if both solutions are members of the same cluster set B_i and $\mathcal{D}_i(x_1) < \mathcal{D}_i(x_2)$ with $\mathcal{D}_i = d_{i,1}(\mathbf{x}) + \theta * d_{i,2}(\mathbf{x})$. Both distances are displayed in Figure 3.11 (Yuan et al., 2014).

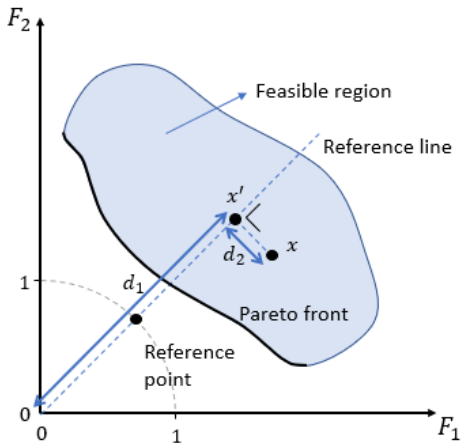


Figure 3.11: Estimation of the convergence of a solution to the Pareto front of a bi-objective problem with objectives F_1 and F_2, using θ-NSGA-III (Yuan et al., 2014).

While Figure 3.11 only considers one solution x, the subscript i of the distances $d_{i,1}$ and $d_{i,2}$ is neglected. Notice that the Pareto front and solutions are not normalised in Figure 3.11 to prevent overcrowding of information. When the solution x is perpendicularly projected on its corresponding reference line, $d_{i,1}$ is defined as the distance between the origin of the (normalised) objective space and this perpendicular projection. $d_{i,2}$ is defined as the Euclidean distance between the solution x and its reference line. θ is a pre-set parameter which can be any number in the interval $]0, 20]$ (Yuan et al., 2014).

The θ-parameter defines the importance attached to distance $d_{i,2}$. If θ equals 0, only the convergence of the solutions to the Pareto front is taken into account. Then no attention is given to the spread of the solutions. Yuan et al. (2014) finds that in this scenario, the θ-NSGA-III algorithm shows the worst results. In general the θ-NSGA-III algorithm shows stable results in a wide range of θ values. Yuan et al. (2014) however opted for $\theta = 5$ in its presented case studies.

3.5 Conclusion

Evolutionary algorithms use biological features such as mutation and reproduction to generate solutions of a multi-objective optimisation problem. Each iteration starts with a parent solution set which is used for the generation of new solutions via crossovers (the reproduction equivalent of the algorithm) and mutations. Evolutionary algorithms show high potential to be used as a solving method for multi-objective optimisation problems. They are able to generate multiple solutions in a single simulation run. Early algorithms however showed a low convergence speed due to their lack of elitism and their performance was highly dependent on a user-defined sharing parameter.

NSGA-II is an elitist evolutionary algorithm that uses crowding distance comparison to maintain a uniform spread in its generated solutions. Solutions are ranked in descending order according to their crowding distance (less crowded solutions are preferred) and in ascending order according to their non-dominated rank (solutions with a low non-dominated rank are preferred). The solutions obtain their non-dominated rank via a fast non-dominated sorting algorithm, which is one of the core features of the NSGA-II algorithm. Eventually, during each iteration, solutions with a high crowding distance and low non-dominated rank are sorted into the highest levels of the solution population and are used as the parent solution set of the following iteration.

NSGA-III is based on the framework of NSGA-II but was especially developed to handle many-objective optimisation problems. These are optimisation problems with four or more objective functions. NSGA-II is unable to generate a sufficient amount of new solutions because with increasing objective functions, an

increasing number of generated solutions are non-dominated. Additionally, the calculation of the crowding distance becomes increasingly difficult when handling many objectives. In NSGA-III it is therefore replaced by a reference points allocation.

In the following part of the thesis text, the materials and methods used during the course of this thesis will be presented together with several case studies. In Chapter 5 the initial NSGA-II and NSGA-III algorithms will be updated to be able to handle constrained optimisation problems. The obtained algorithms will be tested on several case studies, which will simultaneously enable a comparison between both methods and the determination of potential shortcomings.

Part III

Materials and methods

Chapter 4

Numerical methods and software

4.1 Methods used to solve multi-objective optimisation problems

This thesis focusses on the use of evolutionary algorithms to solve multi-objective optimisation problems. In particularly NSGA-II and NSGA-III are extensively used for this purpose. Both algorithms have been explained in detail in the literature study and will therefore be no further discussed in this context.

In the remainder of this thesis text, the NSGA-II and NSGA-III algorithms will be modified so that the algorithms meet the needs of the discussed case studies, which will be presented shortly. Initially the source codes of NSGA-II and NSGA-III will be modified so that they can handle constraints. Furthermore, the algorithms will be analysed on further shortcomings, the eventual aim of which is to alleviate them.

The eventual target is to present improved versions of NSGA-II and NSGA-III that can handle constraints, generate a Pareto front with a trade-off based solution resolution, and make use of a problem-relevant stopping criterion.

4.2 Software

4.2.1 Matlab

The main software package used during the course of this thesis is Matlab R2017a. The Matlab license is provided by the KU Leuven and is for academic use only. Matlab is used to solve the multi-objective optimisation problems presented in this thesis and all additional codes which will be presented in the remainder of this thesis text, are written as functions or scripts in Matlab.

4.2.2 NSGA-II and NSGA-III source codes

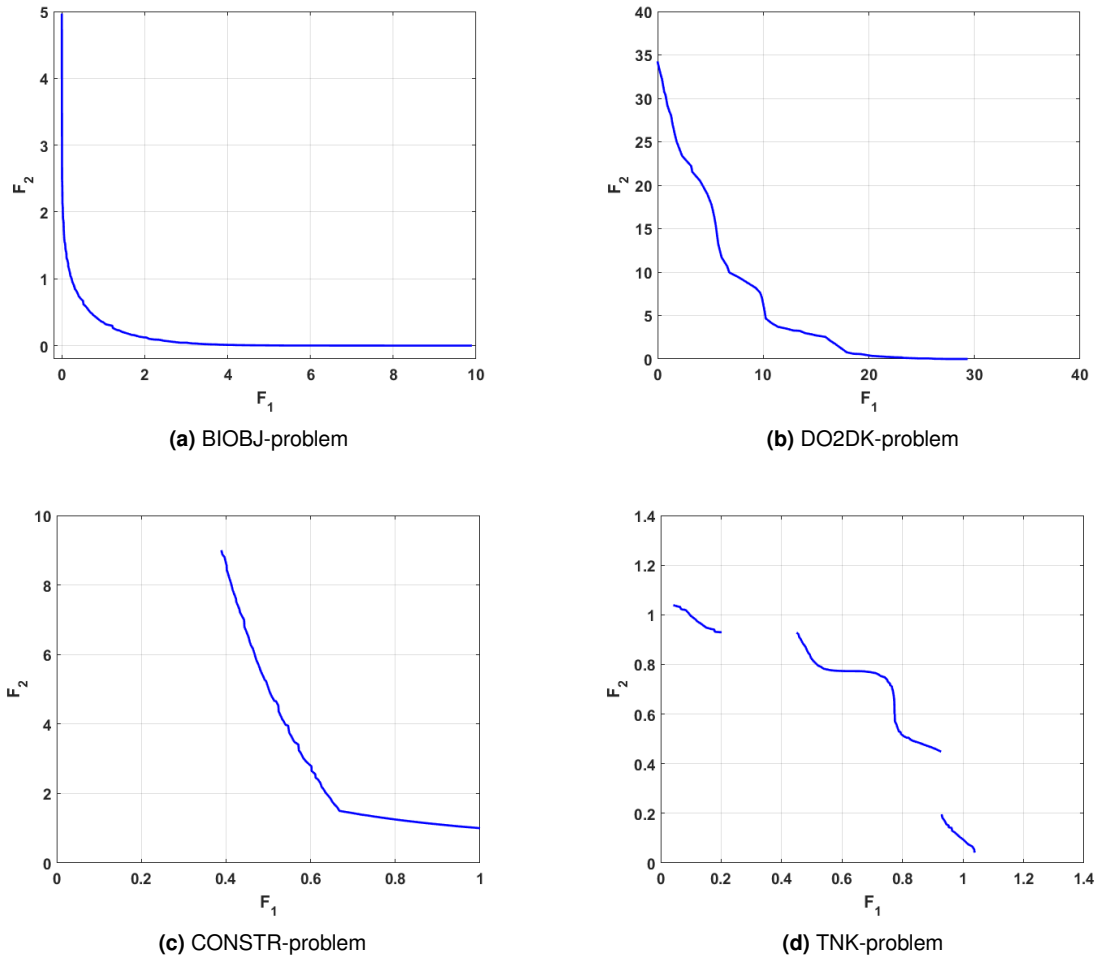
The initial source codes of NSGA-II and NSGA-III are provided by Kalami (2015) and Kalami (2016) respectively. The source codes of the improved versions of NSGA-II and NSGA-III, as presented in the following chapters, are based on the initial source code of the corresponding algorithm. The basic layout of the source codes will largely be kept unchanged. The required additional algorithm properties will be added in the form of Matlab functions to the existing source codes. The source codes will furthermore be simplified and made more accessible for novice Matlab users.

4.3 Numerical case studies

To test the performance of the developed algorithms, several case studies are needed. Six case studies are selected, in order to introduce enough diversity in the tackled optimisation problems. Four bi-objective case studies, one three-objective case study, and one many-objective case study with 5 objectives are used. The case studies are mathematically presented in the following paragraphs, together with their corresponding benchmark Pareto fronts.

4.3.1 Bi-objective case studies

The benchmark Pareto fronts of the bi-objective case studies are represented in Figure 4.1. The corresponding mathematical definitions are presented in the following paragraphs.



(a) BIOBJ-problem

(b) DO2DK-problem

(c) CONSTR-problem

(d) TNK-problem

Figure 4.1: Benchmark Pareto fronts of the bi-objective case studies with their respective objectives F_1 and F_2.

Numerical bi-objective problem

The numerical bi-objective problem (BIOBJ) is provided by Hashem et al. (2017) and is based on a problem defined by Mattson et al. (2014). The problem is mathematically formulated as follows:

$$\min_{\mathbf{z}}(F_1, F_2) \tag{4.1}$$

with

$$F_i = z_i,\ i = 1,2 \tag{4.2}$$

and

$$\left(\frac{z_1 - 10}{10}\right)^8 + \left(\frac{z_2 - 5}{5}\right)^8 - 1 \leq 0 \tag{4.3}$$

$$z_1 \in [-10, 10] \tag{4.4}$$

$$z_2 \in [-10, 10] \tag{4.5}$$

The Pareto front has a sharp curve in the origin area of the objective space and a long and flat regions for higher objective values (see Figure 4.1(a)). The sharp curve represents a high trade-off region while the flat areas represent low trade-off regions.

DO2DK-problem

The DO2DK-problem is provided by Hashem et al. (2017) and is based on a problem described by Branke et al. (2004). It is developed to test (genetic) algorithms on their ability to generate a Pareto front that has a certain skewness and a number of high trade-off knees. The skewness of the DO2DK-problem can be altered via the variable s. The amount of knees can be altered via the variable k. The variable n represents the number of decision variables. The variables s, n, and k are the so-called additional DO2DK variables. The DO2DK-problem is mathematically formulated as follows:

$$\min_{\mathbf{z}}(F_1, F_2) \tag{4.6}$$

with

$$F_1(\mathbf{z}) = g(\mathbf{z}) r(z_1) \sin\left(\pi \frac{z_1}{2^{s+1}} + \left(1 + \frac{2^s - 1}{2^{s+2}}\pi\right) + 1\right) \tag{4.7}$$

$$F_2(\mathbf{z}) = g(\mathbf{z}) r(z_1) \left(\cos\left(\pi \frac{z_1}{2} + \pi\right) + 1\right) \tag{4.8}$$

$$\tag{4.9}$$

and

$$g(\mathbf{z}) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} z_i \tag{4.10}$$

$$r(z_1) = 5 + 10(z_1 - 0.5)^2 + \frac{1}{k}\cos(2k\pi z_1) 2^{s/2} \tag{4.11}$$

$$z_i \in [0,1],\ i = 1,2,\ldots,n \tag{4.12}$$

The additional DO2DK variables of the benchmark Pareto front, as presented in Figure 4.1(b), are set as follows: $s = 1.00$, $n = 300$, $k = 4$. The Pareto front indeed shows four knees or high trade-off areas.

CONSTR-problem

The CONSTR-problem is provided by Deb et al. (2002). It is mathematically formulated as follows:

$$\min_{\mathbf{z}}(F_1, F_2) \tag{4.13}$$

with

$$F_1(\mathbf{z}) = z_1 \tag{4.14}$$
$$F_2(\mathbf{z}) = (1+z_2)/z_1 \tag{4.15}$$

subject to

$$g_1(\mathbf{z}) = z_2 + 9z_1 \geq 6 \tag{4.16}$$
$$g_2(\mathbf{z}) = -z_2 + 9z_1 \geq 1 \tag{4.17}$$

and

$$z_1 \in [0.1, 1.0] \tag{4.18}$$
$$z_2 \in [0, 5] \tag{4.19}$$

The Pareto front of the CONSTR-problem is characterised by a steep section, preceded by a sharp knee and a flat region (see Figure 4.1(c)). Although this problem is comparable to the numerical bi-objective problem, the CONSTR-problem is also included in the case studies because it was observed that the constrained NSGA-II and NSGA-III algorithms deliver better representations of the Pareto front of this problem than of that of the numerical bi-objective problem (see Chapter 5). To be able to compare the functionalities of the constrained NSGA-II and NSGA-III algorithms and the improved versions of both algorithms, it is important to dispose of case studies for which the initial algorithms also show good convergence and solution diversity.

TNK-problem

The TNK-problem is provided by Deb et al. (2002) and is based on a problem formulated by Tanaka et al. (1995). It is mathematically formulated as follows:

$$\min_{\mathbf{z}}(F_1, F_2) \tag{4.20}$$

with

$$F_j = z_j, \; j = 1, 2 \tag{4.21}$$

subject to

$$g_1(\mathbf{z}) = -z_1^2 - z_2^2 + 1 + 0.1\cos(16\arctan(z_1/z_2)) \leq 0 \tag{4.22}$$
$$g_2(\mathbf{z}) = (z_2 - 0.5)^2 + (z_1 - 0.5)^2 \leq 0.5 \tag{4.23}$$

and

$$z_j \in [0, \pi], \; j = 1, 2 \tag{4.24}$$

The Pareto front is represented in Figure 4.1(d). The Pareto front contains several discontinuities, which represent additional difficulties for the genetic algorithms.

4.3.2 Three-objective case study

The selected numerical three-objective case study is the three-objective variant of the DTLZ2-problem, described by Deb et al. (2005). This is a scalable case study and can be adjusted to contain a desired amount of objective functions. In the context of a three-objective case study, the case study is scaled

to the so-called DTLZ2.3-problem. The DTLZ2.3-problem is mathematically formulated as follows (Deb et al., 2005):

$$\min_{\mathbf{z}}(F_1, F_2, F_3) \tag{4.25}$$

with

$$F_1 = (1 + g(\mathbf{z})) \cos\left(z_1 \frac{\pi}{2}\right) \cos\left(z_2 \frac{\pi}{2}\right) \tag{4.26}$$

$$F_2 = (1 + g(\mathbf{z})) \cos\left(z_1 \frac{\pi}{2}\right) \sin\left(z_2 \frac{\pi}{2}\right) \tag{4.27}$$

$$F_3 = (1 + g(\mathbf{z})) \sin\left(z_1 \frac{\pi}{2}\right) \tag{4.28}$$

and

$$g(\mathbf{z}) = \sum_{i=3}^{12} (z_i - 0.5)^2 \tag{4.29}$$

$$z_i \in [0,1],\ \forall i \in \{1, 2, \ldots, 12\} \tag{4.30}$$

The DTLZ2.3-problem consists out of 3 objectives F_i and 12 decision variables z_j. Pareto-optimal solutions are located on the sphere segment with a radius of 1 and as centre the origin of the objective space. The benchmark Pareto front of the DTLZ2.3-problem is represented in Figure 4.2.

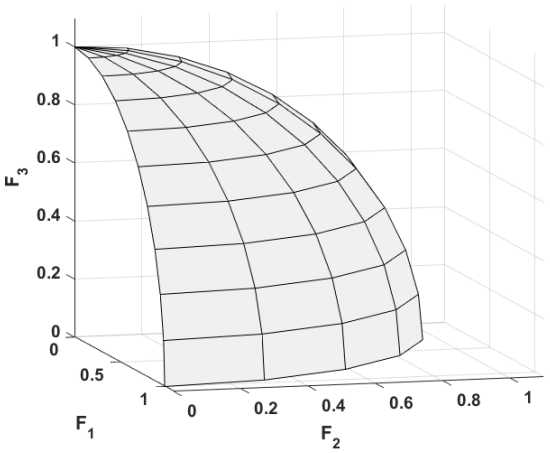


Figure 4.2: Benchmark Pareto front of the three-objective DTLZ2.3-problem, with its corresponding objectives F_1, F_2, and F_2.

4.3.3 Many-objective case study

NSGA-III was specifically developed to handle many-objective optimisation problems ($M > 4$). NSGA-II and NSGA-III should perform similarly on the case studies that have been presented so far. In order to thoroughly evaluate the performance of NSGA-III, a many-objective case study should be included. It is opted to scale up the DTLZ2-problem of Deb et al. (2005) to 5 objectives, rendering the DTLZ2.5-problem. The problem is mathematically formulated as follows:

$$\min_{\mathbf{z}}(F_1, \ldots, F_5) \tag{4.31}$$

with

$$F_1 = (1+g(\mathbf{z}))\cos\left(z_1\frac{\pi}{2}\right)\cos\left(z_2\frac{\pi}{2}\right)\cos\left(z_3\frac{\pi}{2}\right)\cos\left(z_4\frac{\pi}{2}\right) \tag{4.32}$$

$$F_2 = (1+g(\mathbf{z}))\cos\left(z_1\frac{\pi}{2}\right)\cos\left(z_2\frac{\pi}{2}\right)\cos\left(z_3\frac{\pi}{2}\right)\sin\left(z_4\frac{\pi}{2}\right) \tag{4.33}$$

$$F_3 = (1+g(\mathbf{z}))\cos\left(z_1\frac{\pi}{2}\right)\cos\left(z_2\frac{\pi}{2}\right)\sin\left(z_3\frac{\pi}{2}\right) \tag{4.34}$$

$$F_4 = (1+g(\mathbf{z}))\cos\left(z_1\frac{\pi}{2}\right)\sin\left(z_2\frac{\pi}{2}\right) \tag{4.35}$$

$$F_5 = (1+g(\mathbf{z}))\sin\left(z_1\frac{\pi}{2}\right) \tag{4.36}$$

and

$$g(\mathbf{z}) = \sum_{i=5}^{14}(z_i - 0.5)^2 \tag{4.37}$$

$$z_i \in [0,1],\ \forall i \in \{1,2,\ldots,14\} \tag{4.38}$$

The DTLZ2.5-problem consists out of 14 decision variables z_i and 5 objectives F_j.

4.4 Performance parameters

To be able to evaluate the convergence, the solution diversity and the overall performance of the different algorithms, several performance parameter or performance metrics are calculated. These metrics are used to quantify certain properties of the algorithms in order to objectively compare them. These parameters only have a significance if they are related to a multi-objective optimisation problem and the used evolutionary algorithm (Asefi et al., 2014; Rabiee et al., 2012).

4.4.1 Fraction of Pareto-optimal solutions

The fraction of Pareto-optimal solutions (FPOS) is the ratio of the number of generated non-dominated solutions to the total number of generated solutions. If N is the total number of generated solutions and N_{nd} is the number of generated non-dominated solutions, then FPOS is defined as (Asefi et al., 2014):

$$FPOS = \frac{N_{nd}}{N} \tag{4.39}$$

FPOS is a number between 0 and 1 and gives a qualitative evaluation on the performance of the concerned algorithm. The more FPOS tends to converge to 1, the higher the overall performance of the algorithm (Asefi et al., 2014). To evaluate the consistency of the algorithms, the FPOS will be determined as the average FPOS value of 10 repetitions of the same case study with the same algorithm.

4.4.2 Mean ideal distance

The mean ideal distance (MID) is a performance metric which quantifies the convergence of the non-dominated solutions to a pre-defined ideal point c. The MID of a certain problem and algorithm is defined as (Asefi et al., 2014; Rabiee et al., 2012):

$$MID = \frac{\sum_{i=1}^{N_{nd}}\sqrt{\sum_{j=1}^{M}(F_j(i)-F_j(c))^2}}{N_{nd}} \tag{4.40}$$

In Equation (4.40), N_{nd} is the number of non-dominated solutions and F_j is the j-th objective with $j = 1,..,M$. In general, the Utopia point as defined in Equation (2.8) is defined as the ideal point c. If the objective space is normalised via (3.7), then the Utopia is translated to the origin of the normalised objective space. In this scenario, Equation (4.40) can be simplified to (Rabiee et al., 2012):

$$MID = \frac{\sum_{i=1}^{N_{nd}} \sqrt{\sum_{j=1}^{M} F_j(i)^2}}{N_{nd}} \tag{4.41}$$

In order to evaluate the consistency of the algorithms, the MID will be determined as the average MID value, taken over 10 repetitions. The smaller the MID value of a certain problem and algorithm, the higher the convergence of the corresponding generated Pareto-optimal solutions. The MID is namely the mean Euclidean distance between the generated solutions and the ideal, or Utopia point c. Solutions located in the vicinity of the Utopia point are preferred, and those solutions are indicated with a small MID value. Note that, regardless of the normalisation procedure, the MID can have a value higher than 1. This is graphically represented in Figure 4.3 for a bi-objective problem:

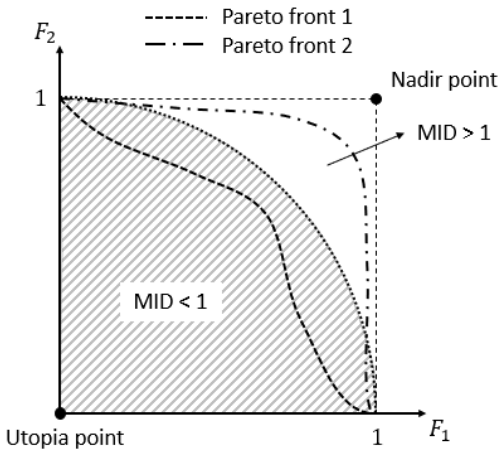


Figure 4.3: Graphical representation of two normalised Pareto fronts and their MID.

If the Pareto front is mainly located within the hypersphere that connects the normalised individual maxima of the objectives, then the MID of the solution population located on this Pareto front, will be smaller than 1. Pareto front 1 in Figure 4.3 is an example of this situation. If the Pareto front is mainly located outside of this hypersphere, like Pareto front 2 in Figure 4.3, then the MID will be greater than 1. The Nadir point is, just like the Utopia point, an extremity of the normalised solution space. The Nadir point contains all the normalised individual maxima of the objectives and is therefore the worst case scenario, while the Utopia point represent the best case scenario. The Matlab code `CalculateMID` is developed to calculate the MID of a solution population and is included in Appendix A.1.

4.4.3 Spread of non-dominated solutions

The MID value of a certain problem and algorithm enables the determination of the overall spread or diversity of the generated solutions. Alongside with the convergence of the solutions to the Pareto front, the obtained solution diversity is an important performance criterion of an evolutionary algorithm. The used diversity performance parameter in this thesis, is the so-called spread of non-dominated solutions (SNDS) (Asefi et al., 2014; Rabiee et al., 2012):

$$SNDS = \sqrt{\frac{\sum_{i=1}^{N_{nd}} (MID - \sum_{j=1}^{M} (F_j(i) - F_j(c)))^2}{N_{nd} - 1}} \tag{4.42}$$

Equation (4.42) represents the general definition of the SNDS of a certain problem and algorithm with N_{nd} the number of non-dominated solutions and F_j the j-th objective with $j = 1,..,M$. The MID value of the concerned algorithm and problem, which is determined via Equation (4.40), must relate to the same ideal point c as used in Equation (4.42). The higher the SNDS value of a certain problem and algorithm, the more diverse the generated non-dominated solutions are. The SNDS value is namely the standard deviation of the average Euclidean distance (MID) between the ideal point c and the generated solutions (Asefi et al., 2014). The Matlab code `CalculateSNDS` is developed to calculate the SNDS of a population and is included in Appendix A.2.

Part IV

Results and discussion

Chapter 5

Constrained NSGA-II and NSGA-III

5.1 Introduction

In this chapter the source codes of the NSGA-II and NSGA-III algorithms, respectively provided by Kalami (2015) and Kalami (2016), are updated to enable them to handle constrained optimisation problems. This is accomplished via the introduction of two feasibility tests. Firstly, both feasibility tests are presented. The obtained constrained NSGA-II and constrained NSGA-III algorithms are subsequently presented. Both algorithms are eventually tested on the case studies that were presented in 4.3, and the acquired results are discussed in detail. The observed shortcomings will finally give rise to the development of the tM- and tDOM-algorithms, which are presented in Chapters 6 and 7 respectively.

5.2 Updating the source code

The source codes of the NSGA-II and -III algorithms were provided by Kalami (2015) and Kalami (2016) respectively. However, both source codes were not suited to handle constraints. Bearing in mind that it is far more common that the solutions of a practical MOOP are subject to constraints, it was decided to extend the existing source codes to the point of being able to handle constrained optimisation problems. For ease of use, the source codes of NSGA-II and NSGA-III were merged into one. This was possible because both algorithms are based on the same framework.

In the literature study, a theoretical constraint handling scheme was presented (see 3.3.7). This scheme involved a new solution domination definition and the addition of a feasibility counter as a new inherent property of a generated solution. The presented constraint handling scheme however does not exclude infeasible solutions. If the number of feasible solutions generated during the t-th iteration is less than N, also infeasible solutions will be included in the eventual solution population P_t. Although the included infeasible solutions violate the imposed constraints by the smallest degree of all the infeasible solutions, they are still practically invalid solutions.

Therefore it is opted to discard all infeasible solutions and to only work with feasible solutions. The constraint handling scheme as presented in 3.3.7 was replaced by two so-called feasibility tests. Both feasibility tests compare the solution to the imposed constraints but return a different output if the solution violates the constraints. The first feasibility test returns a new, feasible solution if the initial solution was found to be infeasible. This type of feasibility test is used to create the random, but feasible, parent solution set P_0 during the initialisation step of the algorithm. The pseudo code of the first feasibility test is represented in Algorithm 3.

Algorithm 3 Feasibility test 1

Require: $\mathbf{p}$, $\mathbf{a}$, $\mathbf{b}$, constraint functions
  $\mathbf{p}' = \mathbf{p}$
  $N = \texttt{size}(\mathbf{p})$
  while $\mathbf{p}$ violates constraint functions **or** $\mathbf{p} < \mathbf{a}$ **or** $\mathbf{p} > \mathbf{b}$ **do**
    $\mathbf{p}' = \texttt{GenerateRandomSolution}(N, \mathbf{a}, \mathbf{b})$
  end while
  return $\mathbf{p}'$

The second feasibility test returns a logical value: logical 0 (or false) if the solution is feasible and logical 1 (or true) if the solution is infeasible and thus violates one or more of the constraints. This second feasibility test is used to verify the feasibility of offspring solutions that are generated during the crossover or mutation steps. It is namely possible that a feasible parent solution is converted into an infeasible offspring solution during both these steps. The logical value is used to regulate a while function: as long as the created solution is infeasible, the mutation or crossover loop is repeated until a feasible solution is generated. The pseudo code of the second feasibility test is represented in Algorithm 4.

Algorithm 4 Feasibility test 2

Require: $\mathbf{p}$, $\mathbf{a}$, $\mathbf{b}$, constraint functions
  if $\mathbf{p}$ violates constraint functions **or** $\mathbf{p} < \mathbf{a}$ **or** $\mathbf{p} > \mathbf{b}$ **then**
    $logic = \texttt{true}$
  else
    $logic = \texttt{false}$
  end if
  return $logic$

The complete constrained NSGA-II and NSGA-III algorithms are represented in Algorithms B.1 and B.2 respectively in Appendix B. The basic framework of the algorithms, as provided by Kalami (2015) and Kalami (2016), is kept unchanged. Only the feasibility tests are added.

5.3 Case studies

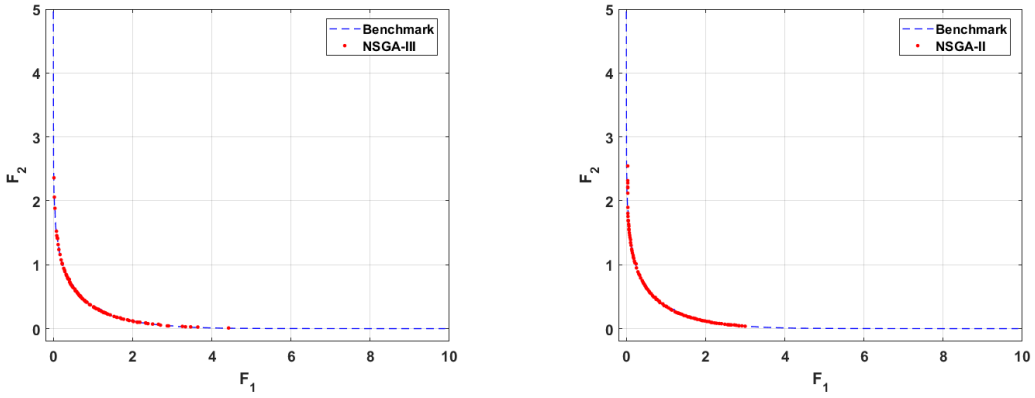
The six case studies as presented in section 4.3 will be used to demonstrate the constrained NSGA-II and NSGA-III algorithms and compare the two algorithms. The algorithms will be evaluated on overall efficiency, convergence and solution diversity via the determination of the performance parameters as presented in section 4.4. The used settings of the algorithm parameters of each case study are summarised in Table 5.1.

Table 5.1: Algorithm parameter settings used for the case studies.

Parameter	Value
Iterations	75
Population size	100
Crossover probability p_c	90.0 %
Mutation probability p_m	10.0 %
Mutation rate μ	5.0 %
Mutation step size σ	$0.05 \times (\mathbf{b} - \mathbf{a})$
Objective subdivisions	99 (*), 13 (**), 5 (***)
Reference points	100 (*), 105 (**), 126 (***)

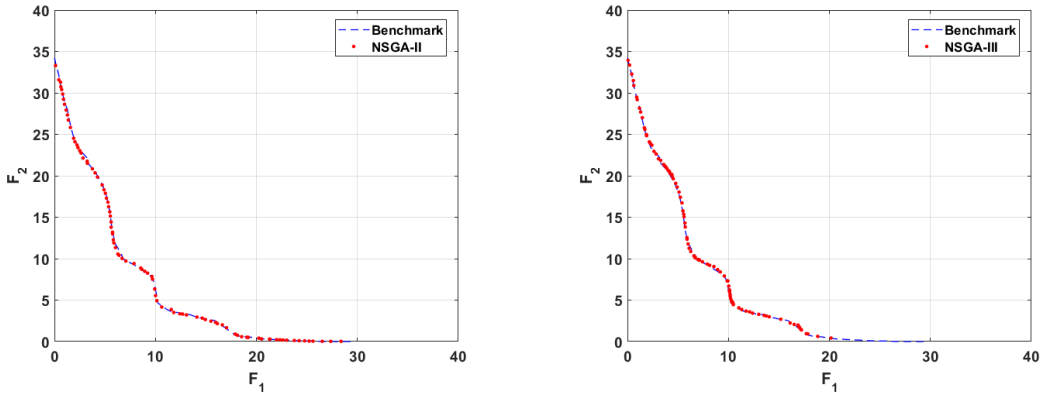
$\mathbf{a}$ and $\mathbf{b}$ are the lower and upper boundary respectively of the process variables. Concerning the number of objectives subdivisions and reference points, (*) refers to the bi-objective case studies, (**) to the three-objective case study and (***) to the many-objective case study. The number of subdivisions have been intentionally chosen so that the number of reference points $H \geq N$.

5.3.1 The bi-objective case studies



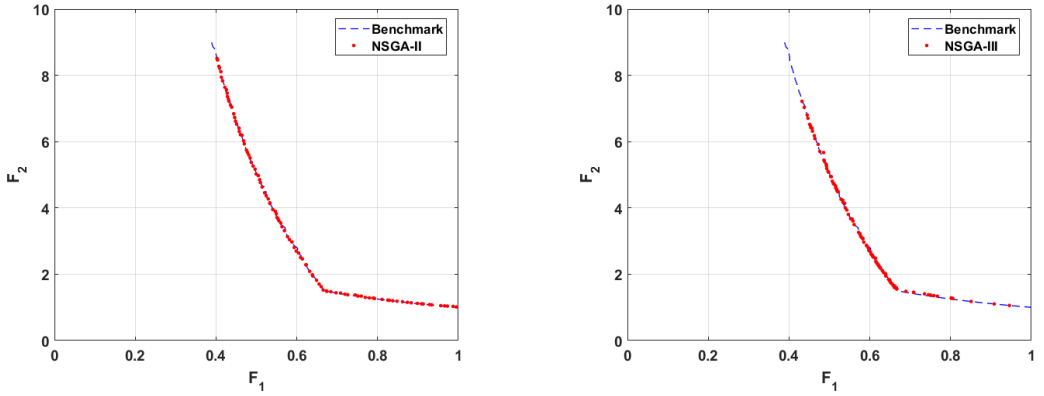
(a) Constr. NSGA-II **(b)** Constr. NSGA-III

Figure 5.1: Pareto front of the BIOBJ-problem generated via constrained NSGA-II(a) and constrained NSGA-III(b).



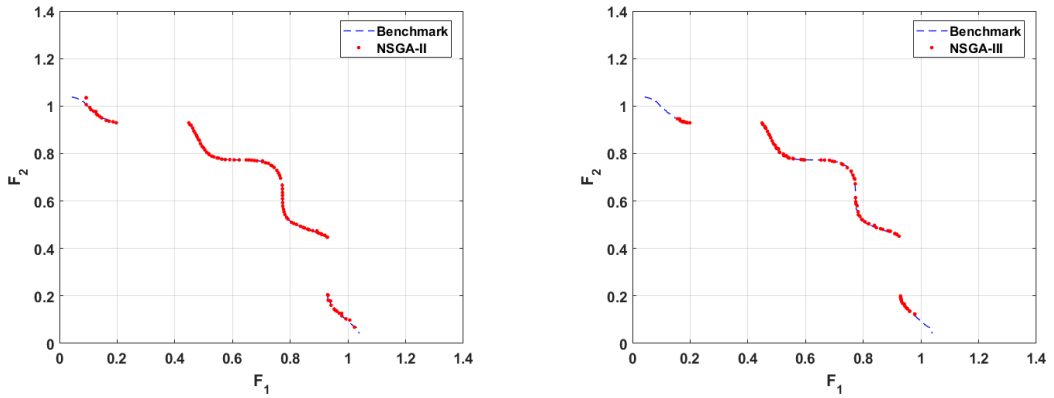
(a) Constr. NSGA-II **(b)** Constr. NSGA-III

Figure 5.2: Pareto front of the DO2DK-problem generated via constrained NSGA-II(a) and constrained NSGA-III(b).



(a) Constr. NSGA-II **(b)** Constr. NSGA-III

Figure 5.3: Pareto front of the CONSTR-problem generated via constrained NSGA-II(a) and constrained NSGA-III(b).



(a) Constr. NSGA-II **(b)** Constr. NSGA-III

Figure 5.4: Pareto front of the TNK-problem generated via constrained NSGA-II(a) and constrained NSGA-III(b).

The Pareto front of the BIOBJ-problem is represented in Figures 5.1(a) and 5.1(b). These Pareto fronts are generated with constrained NSGA-II and constrained NSGA-III respectively. Note that both constrained al-

gorithms do not have a built-in method that enables the algorithm to distinguish between solutions based on their trade-off. Despite the lack of such a method, both the constrained NSGA-II and constrained NSGA-III algorithms tend to generate solutions that are located in the high trade-off area of the Pareto front, i.e. the knee. The solution diversity in both cases is therefore poorly compared to the benchmark. The generated solutions of both algorithms have however, despite the unwanted convergence to the high trade-off area, converged to the benchmark Pareto front. Note that the Pareto fronts represented in Figure 5.1, and all the ensuing figures, are only snapshots of the algorithms performances. These Pareto fronts are namely generated during one repetition of the algorithm and are not based on the mean performances of the algorithm. It was opted to include them nonetheless because several general conclusion and remarks can be based on the representation of the generated Pareto fronts. The performance parameters, represented in Table 5.2, will give a more inclusive overview of the performance of both algorithms.

The Pareto front of the DO2DK-problem is represented in Figures 5.2(a) and 5.2(b). The generated solutions of both algorithms are highly diverse and are spread uniformly on the Pareto front, in contrast to the generated solutions of the BIOBJ-problem. Additionally it is clear that the generated solutions have, for both constrained-algorithms, converged to the benchmark DO2DK Pareto front.

Figures 5.3(a) and 5.3(b) represent the Pareto front of the CONSTR-problem. The CONSTR-problem has, just like the BIOBJ-problem, distinct high and low trade-off regions. Regardless of the high level of similarity, it is found that the constrained NSGA-II and constrained NSGA-III algorithms produce a more diverse solution population for the CONSTR-problem than for the BIOBJ-problem, and thus show a better overall performance. This phenomenon might indicate that the (overall) performances of the two algorithms are problem-dependent. This is naturally an unwanted algorithm property. This problem will be extensively studied in Chapter 6 and Chapter 7, after which an according algorithm improvement will be presented.

Finally, the Pareto fronts of the TNK-problem are represented in Figures 5.4(a) and 5.4(a). This case study was included in order to test the algorithms abilities to handle discontinuous Pareto fronts. Based on Figure 5.4 it can be concluded that both constrained NSGA-II and constrained NSGA-III are capable of handling discontinuous Pareto fronts. In both cases no solutions were generated in the discontinuous areas and the extremities of the continuous parts of the Pareto front are not problematic points. Additionally, all the generated solutions have converged to the benchmark TNK Pareto front.

The necessity to objectively compare and quantify the performance of the two algorithms per problem, calls for the use of the three performance parameters FPOS, MID, and SNDS that were presented in 4.4. Both algorithms were repeated ten times for each problem. Table 5.2 display the mean value and standard deviation of the performance parameters, which were calculated after each repetition. The mean values and standard deviations were calculated using the *mean* function and the *std* function of Matlab respectively.

Table 5.2: Performance parameters and runtime of the bi-objective case studies. The best mean value, per case study, of the concerned performance parameter is set in bold.

	Constr. NSGA-II		Constr. NSGA-III	
Performance parameter	**Mean**	**St. dev.**	**Mean**	**St. dev.**
BIOBJ-problem				
FPOS [-]	1.0000	0.0000	1.0000	0.0000
MID [-]	0.4548	0.0604	**0.4377**	**0.0631**
SNDS [-]	**0.1861**	**0.0282**	0.1548	0.0103
Runtime [s]	**217.9184**	**12.0773**	314.2116	6.6605
DO2DK-problem				
FPOS [-]	1.0000	0.0000	1.0000	0.0000
MID [-]	0.6270	0.0108	**0.6230**	**0.0217**
SNDS [-]	**0.1653**	**0.0031**	0.1620	0.0032
Runtime [s]	**233.4890**	**5.4313**	305.4324	12.4681
CONSTR-problem				
FPOS [-]	1.0000	0.0000	1.0000	0.0000
MID [-]	0.6240	0.0112	**0.5795**	**0.0132**
SNDS [-]	0.1693	0.0051	**0.1730**	**0.0107**
Runtime [s]	**200.6040**	**12.3410**	323.3773	31.9706
TNK-problem				
FPOS [-]	1.0000	0.0000	1.0000	0.0000
MID [-]	**0.9417**	**0.0147**	0.9514	0.0088
SNDS [-]	**0.3002**	**0.0111**	0.2726	0.0110
Runtime [s]	**156.0805**	**8.3336**	242.7833	35.7819

Concerning the BIOBJ- and DO2DK-problem, the constrained NSGA-III showed the best convergence and the constrained NSGA-II showed the best solution diversity. For the BIOBJ-problem, the constrained NSGA-III displayed a 3.760 % better convergence and the NSGA-II displayed a 16.82 % better solution diversity. The differences in performance are smaller for the DO2DK-problem and add up to a 0.6380 % and 1.996 % improvement respectively. When considering the CONSTR- and TNK-problem however, a completely different scenario is observed. The constrained NSGA-III algorithm outperforms the constrained NSGA-II algorithm both in convergence and solution diversity in case of the CONSTR-problem, while the opposite scenario is observed in case of the TNK-problem. The constrained NSGA-III algorithm displays a 7.131 % and 2.139 % increased convergence and solution diversity respectively for the CONSTR-problem. In case of the TNK-problem, the constrained NSGA-II algorithm outperforms the constrained NSGA-III algorithm by 1.020 % and 9.194 % on convergence and solution diversity respectively. The inconsistency in which the two algorithms outperform each other can be another indicator that the overall performance of the considered algorithms is problem-dependent.

The fact that the constrained NSGA-II algorithm slightly outperforms the constrained NSGA-III algorithm on solution diversity in case of the BIOBJ- and DO2DK-problem, is a surprise. NSGA-III uses uniformly spread reference points in the objective space in order to achieve sufficient solution diversity. The overall expectation is that the uniform spread of the reference points is eventually mirrored in the solution spread. Nevertheless there are no diversity restriction for the initial, randomly generated, population. This is also the case for the NSGA-II algorithm. If this population is not diverse enough, it will take both algorithms a substantial, and often non-practical, amount of iterations to finally achieve a diverse solution set. The magnitude of this complication is however very problem-dependent. Based on the graphical representation of the Pareto fronts of both problems in Figures 5.1 and 5.2 respectively, it can be concluded that

the problem of the insufficient solution diversity only takes substantial proportions for the BIOBJ-problem and not for the DO2DK-problem. This is also confirmed by the increased relative difference in solution diversity between the BIOBJ- and DO2DK-problem (16.82 % vs. 1.996 %).

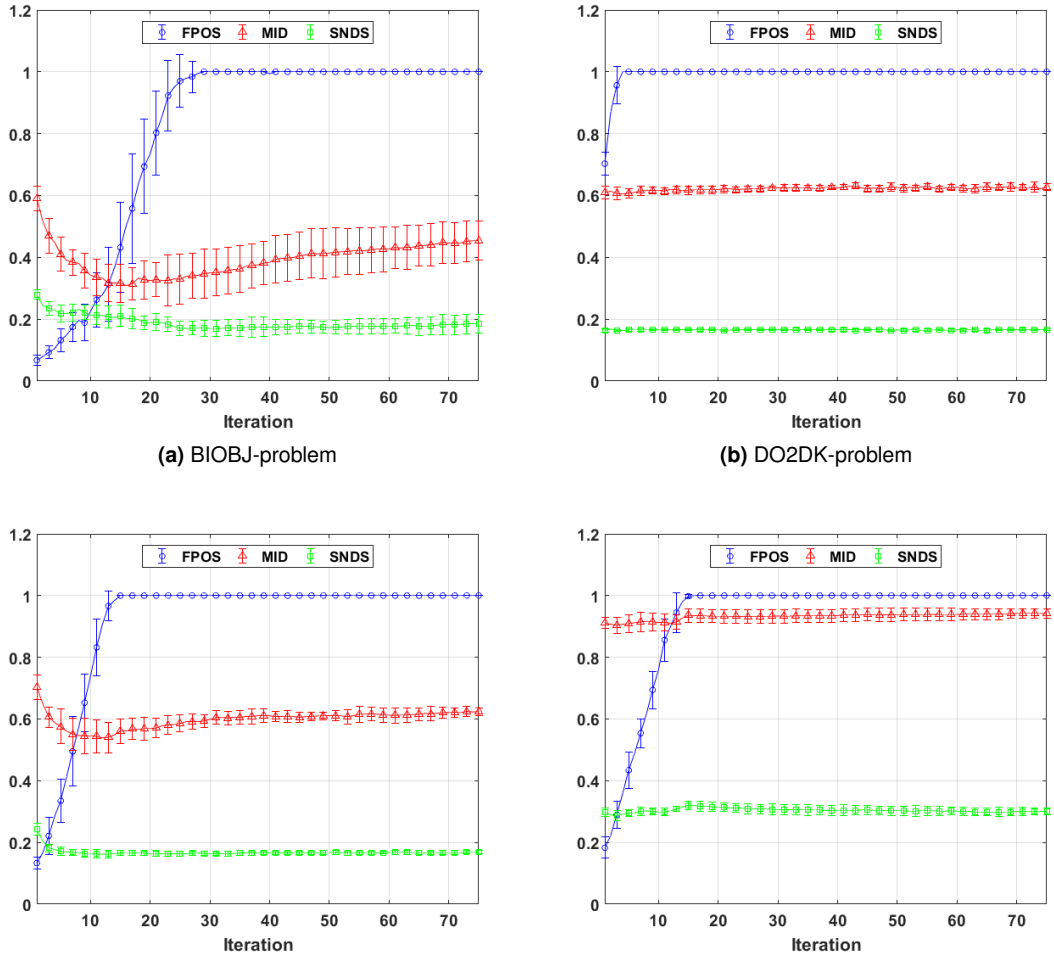
In general it can be concluded that constrained NSGA-II and constrained NSGA-III perform similarly when handling bi-objective problems. Furthermore it can be concluded that the largest performance differences are made on the spread of the generated solutions. The uniform spread of solutions on the Pareto front, still proves to be a difficulty, despite the incorporation of crowding distances or reference points. It should be noted however that the used performance parameters MID and SNDS are not independent from each other. This can be easily understood when Equation (4.42) is considered. The MID value is used to calculate the SNDS value. The two parameters must always be considered simultaneously. A higher MID value does not always indicate a decreased convergence, but can easily be the result of a more diverse solution population. The MID and SNDS performance parameters are mainly selected for their intuitive performance quantification and easy implementation but they do not offer an all-embracing description of the nature of the Pareto front and the performance of the algorithm.

A more practical performance parameter is also included in Table 5.2, being the runtime of the algorithm. It goes without saying that it is desired that the runtime of the algorithms is as minimal as possible, i.a. from a practical point of view. In this context, the constrained NSGA-II algorithm significantly outperforms the constrained NSGA-III algorithm for all four case studies. When handling a small amount of objectives, the allocation of solutions to reference points is probably too cumbersome and is therefore slowing the algorithm down. Note that the extensive use of the non-dominated sorting function and the diversity-enabling functions in both algorithms are also major contributors to the total runtime. For instance when the constrained NSGA-II algorithm is run and the maximum number of iterations equals t, the non-dominated sorting function and the crowding distance function are run $2t + 1$ times each (see Algorithm B.1). Both these functions contain repeated sorting and comparison steps and are the two most time consuming functions of the entire algorithm. In Chapter 6 and Chapter 7 it will be evaluated if, in some cases, the non-dominated sorting function or diversity-enabling function of the considered algorithm, can be left out to save computing time.

Lastly, the most obvious observation that can be made based on Table 5.2, is that for all the bi-objective case studies that were concerned, and for both algorithms, the FPOS performance parameter is always equal to 1. This means that the eventual obtained solution population always entirely consisted out of non-dominated solutions. Moreover it was observed that the maximum number of iterations, i.e. 75, was set too high. The solution population already consisted entirely out of non-dominated solutions after approximately 15 iterations. Subsequently a lot of computation time was spent on unnecessary further iterations. The only valid arguments to continue iterating nonetheless, are on one side the anticipation that the solutions will further converge to the Pareto front, and on the other side the anticipation that the solution diversity will increase.

The first argument however implies that eventually, whilst the iteration process continues, new non-dominated solutions are generated. Because these solutions are more closely located to the Pareto front, it can be expected that they will dominated one or more solutions of the non-dominated solutions of the previous iteration. Thus, if it were the case that the solutions would further converge to Pareto front, it is likely that the number of non-dominated solutions would fluctuate. This is however not the case. Once the solution population consists entirely out of non-dominated solutions, it stays this way until the maximum number of iterations is reached.

In order to estimate the necessity of continued iterating after the maximum number of non-dominated solutions is reached, the performance parameters are additionally calculated during each iteration of the ten repetitions of both algorithms. The mean values and standard deviations of the performance parameters per iteration of the constrained NSGA-II algorithm are graphically displayed in Figure 5.5. These are the so-called performance plots. The corresponding performance plots of the NSGA-II algorithm are included in Appendix B.1. The conclusion about constrained NSGA-II algorithm, drawn based on Figure 5.5, can be extrapolated to the constrained NSGA-III algorithm.



(a) BIOBJ-problem

(b) DO2DK-problem

(c) CONSTR-problem

(d) TNK-problem

Figure 5.5: Performance plot of the constrained NSGA-II algorithm in case of the bi-objective problems (average taken over 10 repetitions).

Figure 5.5 clearly displays that once the FPOS equals 1, this value does not change when the iteration process continues. The number of iterations needed to obtain the maximum FPOS value of 1, is highly problem dependent. When the CONSTR- and TNK-problem are concerned, it can be seen that these two problems need approximately the same number of iterations ($\pm$ 15) before FPOS reaches its maximum value. The DO2DK-problem needs far less iterations ($\pm$ 2), and the BIOBJ-problem needs more ($\pm$ 30).

Additionally it can be seen that the average MID-values of the BIOBJ- and CONSTR-problem increase after the FPOS has reached its maximum value and the iteration process continues. The average MID-

values of the DO2DK- and TNK-problem on the other hand reach a constant value after FPOS has reached its maximum value and the algorithm is not stopped. An increase in MID means that the average Euclidean distance between the solutions and the Utopia point increases, i.e. the solutions move away from the Utopia point. This is obviously an unwanted scenario because this means that convergence of the solutions to the Pareto front diminishes if the algorithm is not stopped. As said before, an increase in MID can also indicate that the solution population becomes more diverse. However, for both the BIOBJ- and CONSTR-problem, Figure 5.5 indicates that the SNDS stabilises (or even decreases) for both case studies after the FPOS reaches its maximum value. Therefore it can be concluded that the diversification of the solution population is here not the reason why the MID increases.

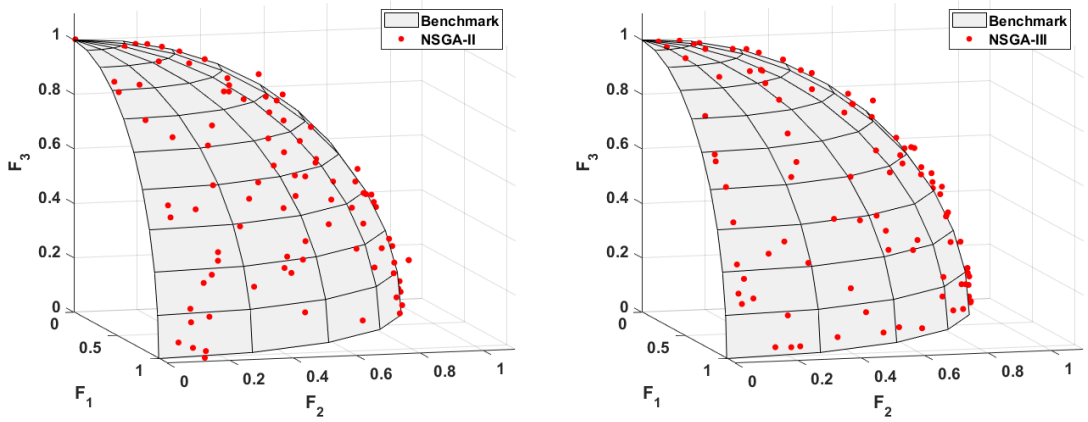
A decrease in convergence however, is an unexpected phenomenon. NSGA-II and NSGA-III are elitist algorithms and thus the best solutions of the previous iteration are kept unchanged in the current one. A decrease in convergence implies that the selected solutions of the current iteration are, on average, located further away from the Pareto front than the selected solutions of during the previous iteration. This is in contradiction to the elitist basis of the algorithms because solutions that display a decreased convergence are by definition worse than solutions with a higher convergence. This observed anomaly is most likely the result of a flawed solution sorting and selecting procedure. The non-dominated rank of a solution is emphasised too strongly when selecting N solutions from the combined solution set R_t (see 3.3.6) if $|\mathcal{F}_1| < N$. If however $|\mathcal{F}_1| \geq N$, then the crowding distance is emphasised too strongly, resulting in the preference of diverse solutions with a lower convergence, whilst remaining non-dominated.

For the other two case studies, the DO2DK- and TNK-problem, there is no decline or improvement detected in the convergence of the solutions. This observations also renders further iterating futile because of the lack of improvement. The risk of potentially moving away from the Pareto front is an additional incentive to stop the algorithm when the FPOS reaches its maximum value of 1. It can therefore be concluded that the first argument to continue iterating, i.e. the anticipation of a better convergence, is invalid based on the observations made above.

The second argument to continue the iteration process after the FPOS has reached its maximum value, i.e. the anticipation that the solution diversity will increase, is also proven invalid based on Figure 5.5. An increase in solution diversity is indicated by an increase in the SNDS-value. However, it can be seen in the case of the BIOBJ- and CONSTR-problem, that continued iterating results in a decrease in the SNDS-value. These two problems also show an increase in the MID-value when iterating continues. Solutions thus move away from the Utopia point, or Pareto front, and become less diverse. This is a worst case scenario and has to be avoided. Considering the other two case studies: for the DO2DK-problem it can be seen that the SNDS stabilises when the iteration process is continued. Only the TNK-problem displays a small increase in its SNDS-value.

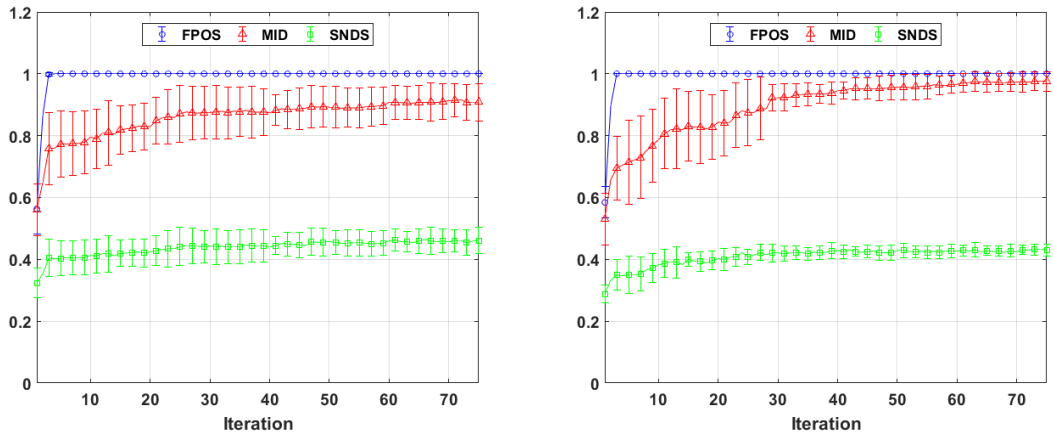
The general trend when the iteration process is continued after the FPOS reaches its maximum value, is that solutions move away from the Pareto front and become less diverse, or that the situation stabilises and no improvement or decline is seen. Based on these two remarks, it is concluded that further iterating after the FPOS has reached its maximum value is futile and should be avoided.

5.3.2 The three-objective case study



(a) Constr. NSGA-II **(b)** Constr. NSGA-III

Figure 5.6: Pareto fronts of the DTLZ2.3-problem, generated via constrained NSGA-II (a) and constrained NSGA-III (b).



(a) Constr. NSGA-II **(b)** Constr. NSGA-III

Figure 5.7: Performance plot of the constrained NSGA-II and NSGA-III algorithms in case of the DTLZ2.3-problem (average taken over 10 repetitions).

The Pareto fronts of the DTLZ2.3-problem are represented in Figure 5.6. The Pareto fronts are generated with constrained NSGA-II and constrained NSGA-III with 105 reference points respectively. The corresponding performance plots are represented in Figure 5.7. The final values of the performance parameters are included in Table 5.3.

Table 5.3: Performance parameters and runtime of the DTLZ2.3-problem. The best value of the concerned performance parameter is set in bold.

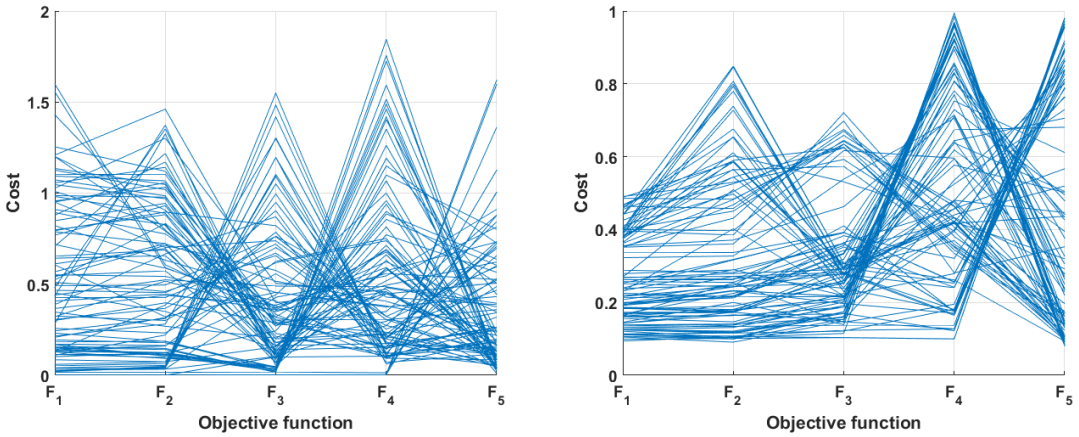
	Constr. NSGA-II		Constr. NSGA-III	
Performance parameter	**Mean**	**St. dev.**	**Mean**	**St. dev.**
FPOS [-]	1.0000	0.0000	1.0000	0.0000
MID [-]	**0.9081**	**0.0575**	0.9756	0.0295
SNDS [-]	**0.4604**	**0.0407**	0.4303	0.0181
Runtime [s]	**217.4329**	**17.9190**	301.1893	23.6377

Figure 5.7 displays that the FPOS quickly reaches its maximum value. Just like for the bi-objective DO2DK-problem, only $\pm$ 2 iterations are needed to obtain a solution population that completely consists out of non-dominated solutions. The MID keeps increasing, but this is accompanied with an increase in SNDS while FPOS$<$1. The early increase in MID is therefore most likely the result of the diversification of the solution population. However, when the FPOS$=$1 and the iteration process is continued, the MID continues to increase but the SNDS tends to stabilise. At this point, the increase in MID is no longer the result of the diversification of the solution population but a result of the diminishing convergence of the solutions.

Based on the performance parameters displayed in Table 5.3, it can be concluded that the constrained NSGA-II algorithm outperforms the constrained NSGA-III algorithm on convergence, solution diversity, and runtime. The failing performance of the constrained NSGA-III algorithm is again most likely due to the cumbersome allocation of solutions to the reference points.

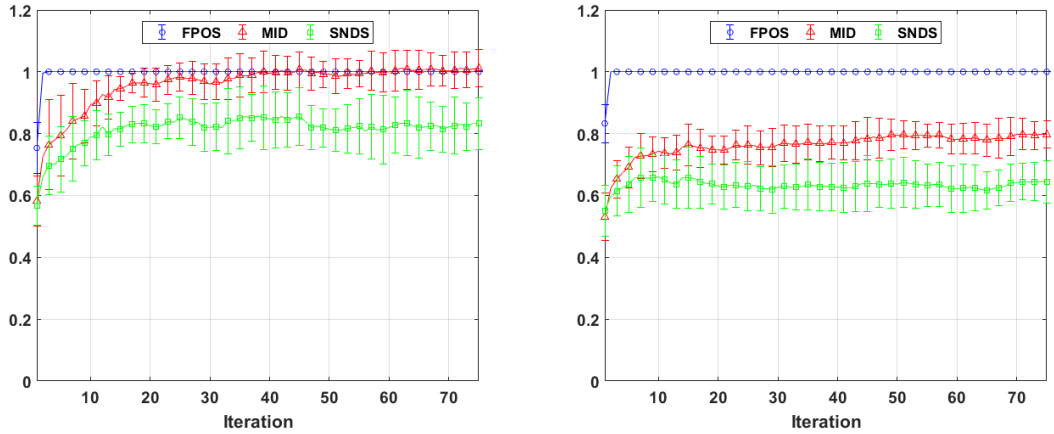
5.3.3 The many-objective case study

In order to fully exploit and test the strength of the constrained NSGA-III algorithm, the DTLZ2.5-problem is included which is a many-objective optimisation problem consisting out of 5 objective functions (see 4.3.3). The Pareto front of the DTLZ2.5-problem is represented in Figure 5.8 via the use of a parallel-coordinate plot. The objective costs of each generated solution are plotted on a different, parallel axis. The plots corresponding to the same solution are subsequently connected. This technique enables the user to graphically represent high-dimensional data. The performance plots are displayed in Figure 5.9.



(a) Constr. NSGA-II **(b)** Constr. NSGA-III

Figure 5.8: Graphical representation of the Pareto front of the DTLZ2.5-problem via parallel-coordinates plots.



(a) Constr. NSGA-II **(b)** Constr. NSGA-III

Figure 5.9: Performance plot of the constrained NSGA-II and NSGA-III algorithms in case of the DTLZ2.5-problem (mean taken over 10 repetitions).

Table 5.4 summarises the final mean values of the performance parameters and the mean runtime of both constrained algorithms.

Table 5.4: Performance parameters and runtime of the DTLZ2.5-problem. The best values of the concerned performance parameter is set in bold.

	Constr. NSGA-II		**Constr. NSGA-III**	
Performance parameter	**Mean**	**St. dev.**	**Mean**	**St. dev.**
FPOS [-]	1.0000	0.0000	1.0000	0.0000
MID [-]	**0.7976**	**0.0412**	1.0105	0.0730
SNDS [-]	0.6427	0.0650	**0.8159**	**0.0988**
Time per loop [s]	**231.8878**	**3.2028**	325.4437	11.0706

Based on Figure 5.8, a first conclusion can be drawn about the convergence of the solutions. On average, the costs of all the objective functions will be lower for the solutions generated via constrained NSGA-II than for those generated via constrained NSGA-III. This observation is also confirmed by Figure 5.9 and the MID values displayed in Table 5.4. However it can be seen in Figure 5.9(b) that for the first 20 iterations, the MID and SNDS increase in a similar manner. The increase in MID is in this scenario thus more likely the result of the diversification of the solution population. This is confirmed by the simultaneous increase in SNDS. The subsequent increase in MID is on the other hand more likely the result of a decrease in convergence while the SNDS has stabilised at this point. For the constrained NSGA-II algorithm, a similar, but less noticeable, phenomenon can be seen.

Again it is concluded that the unnecessary continuation of the iteration process is detrimental for the performance of the algorithm. In this instance, especially solution convergence is negatively affected by the continuation of the iteration process. This case study is nonetheless the first case study for which limited continued iterating, after the FPOS reached its maximum value, proved to be beneficial for the solution diversity. To conclude it is again found that the constrained NSGA-III algorithm is significantly more time consuming than its NSGA-II equivalent.

The main doubt regarding this case study is how representative many-objective case studies are for real-

life optimisation problems. The decision maker will often prefer a 2D or 3D representation of the Pareto front. This would enable him/her to facilitate the decision making process. The artificial Pareto front representation of many-objective optimisation problems by means of parallel coordinates, is not an intuitive representation to work with. It is for instance more difficult to distinguish between multiple solutions and to establish how different solutions are related to each other. If an optimisation problem requires many objectives, it is more appropriate to ascertain if all the concerned objectives are equally important. If not, an objective reduction is recommended so that the objective space is reduced to a two- or three-dimensional space.

Objective reduction however offers a complete new set of challenges which will not be further discussed in this thesis. Based on the low added value of a many-objective optimisation problem, it is opted to not further examine this case study.

5.4 Conclusion

With regard to enabling the source codes of the NSGA-II and NSGA-III algorithm (provided by Kalami (2015) and Kalami (2016) respectively) to handle constraints, two feasibility test were introduced in the onset of this chapter. The subsequently acquired constrained algorithms are included in Appendixes B.1 and B.2. On this point, the basic framework of the algorithms was kept unchanged. Only the feasibility tests were added. The constrained algorithms were eventually tested on the case studies that were presented in 4.3. The performance parameters, as presented in 4.4, and the mean runtime of the algorithm, were used to quantify the algorithm's performance. The most important observations that were made are the following:

1. The overall performance of the algorithms was often highly problem-dependent. For instance both algorithms displayed a poor solution diversity for the BIOBJ-problem, but a high solution diversity for the DO2DK-problem. Despite the use of diversity-enabling function (the crowding distance or the use of reference points), solutions tended to converge towards local optima rendering an non-uniform solution spread on the Pareto front. Also the runtime of the algorithm was highly problem-dependent. The extensive use of the time consuming non-dominated sorting and diversity-enabling functions, were also a main reason for this.

2. As a result of a naive stopping criterion (reaching a pre-defined number of iterations), a lot of computational time was uselessly spend on the continuation of the iterating process, while the fraction of generated non-dominated solutions already equalled 1. The only two valid arguments to continue iterating nonetheless, were proven invalid. The first argument is that continuing the iteration process would increase the convergence of the solutions to the Pareto front. In practice it was seen that in most cases the opposite scenario was observed, or in the best case, no change at all. The second argument to continue the iterating process, is the anticipation that the solution diversity will increase. Again, often the opposite scenario was observed.

3. Constrained NSGA-III, although specifically developed to handle many-objective optimisation problems, was more often than not unable to press home its built-in advantages over the constrained NSGA-II algorithm. The convergence of the solutions generated via constrained NSGA-III diminished with an increasing number of objective functions and the solution diversity was only for two case studies (CONSTR-problem and DTLZ2.5-problem) higher than the solution diversity of the constrained NSGA-II algorithm. The most striking observation regarding constrained NSGA-III was that its runtime was always significantly higher than the runtime of constrained NSGA-II for the corresponding problem. The allocation of solutions to the reference points is probably cumbersome, slowing the algorithm down.

An additional major shortcoming of both algorithms is that they are unable to distinguish between solutions based on their trade-off. A sufficient solution diversity, with an emphasis on high trade-off solutions is desired. All these observed shortcomings form the basics on which the tM- and tDOM-algorithms are based. Both types of algorithms are presented in Chapter 6 and Chapter 7 respectively.

Chapter 6

tM-NSGA-II and tM-NSGA-III

6.1 Introduction

While testing the constrained versions of NSGA-II and NSGA-III on the case studies presented in 4.3, three major shortcomings of the algorithms surfaced. The NSGA-III algorithm was found to be slow and was in many instances outperformed by the more intuitive NSGA-II algorithm. Also, the performance of both algorithms was highly problem-dependent. The same algorithm for instance generated solutions with a poor diversity for one problem, but a highly diverse solution set for another problem. Especially the BIOBJ-problem proved difficult in this respect.

The naive stopping criterion was another major shortcoming of both algorithms. Up until now, the algorithms were stopped when a pre-defined number of maximum iterations is reached. While this value is arbitrary chosen and has no relevance to the concerned optimisation problem, it resulted in the waste of computation time. In all the tested case studies, only a small fraction of the pre-defined iterations was needed to obtain a solution population that completely consisted out of non-dominated solutions and thus for FPOS to reach its maximum value 1. Continuing the iteration process after this point could be beneficial if the convergence and/or the diversity of the generated solutions increased. These two arguments were proven to be invalid and in several case studies it was even found that continuing the iteration process was detrimental with respect to those two properties.

The above shortcomings will be remedied by the implementation of a problem-relevant stopping criterion and the simplification of both algorithms. For instance the extensive use of time consuming functions, like the non-dominated sorting function, will be restricted. Additionally a trade-off function will be introduced, enabling the algorithm to emphasise high trade-off solutions over low trade-off solutions. In this chapter, tM-NSGA-II and tM-NSGA-III will be presented. 't' refers to the built-in trade-off functions, and 'M' refers to the novel stopping criterion. In Chapter 7, tDOM-NSGA-II and tDOM-NSGA-III will be presented. The same built-in trade-off function is included in these algorithms to emphasise high trade-off solutions, but in these algorithms, the trade-off of solutions is also used as a stopping criterion.

The built up of this chapter is the analogous of that of the previous one: firstly the new algorithm functions will be presented, in this case the trade-off function and the novel stopping criterion. Subsequently the complete tM-algorithms will be presented in detail. Eventually both algorithms will be tested on the case studies that were presented in 4.3 and the obtained results will be discussed and compared to those of the constrained algorithms.

6.2 Trade-off as a second crowding distance

Pareto fronts are generated to supply multiple equivalent solutions of an optimisation problem. The solutions are equivalent in the way that all the solutions minimise the concerned optimisation problem, but they do so in a different manner. The decision maker has to choose one solution from the generated solution population and this can prove to be difficult. Mattson et al. (2014) states that it can be advantageous to reduce the solution population to a smaller population which only contains solutions with a high probability of being selected by the decision maker. The decision maker's main interest are namely solutions that display a certain trade-off Δt and distribution Δr.

The decision maker can specify his/her preferred solutions before the solution population is generated (a priori decision making), yet this scenario is fairly uncommon. The decision maker often does not dispose of the information enabling him/her to select his/her preferred solutions upfront. The most common approach to reduce the size of a solution population is by filtering preferred solution from an initially large solution population (a posteriori decision making), based on the decision maker's desired solution trade-off Δt and distribution Δr. It is this approach that will be included in the tM- and tDOM-algorithms.

The main disadvantage of a posteriori decision making however is that this method requires the generation of a large number of solutions, of which a substantial fraction is discarded at the end of the filtering process. Discarding those solutions simultaneously implies that a large amount of computing time is used to generate useless solutions. Hashem et al. (2017) introduced a divide and conquer scheme, enabling deterministic algorithms, like NBI (see 2.3.2), to only generate solutions that meet the minimal required trade-off Δt and distribution Δr. Solutions are generated at a rate of one solution per iteration via a recursive scheme. If a generated solution does not meet the minimal required trade-off Δt or distribution Δr any longer, the algorithm is stopped or is urged to stop exploring the concerned area of the Pareto front. The trade-off Δt and distribution Δr of solutions are thus additionally used as a stopping criterion. The fact that stochastic algorithms generate multiple solutions per iteration causes the divide and conquer scheme of Hashem et al. (2017) to be unsuitable for evolutionary algorithms like NSGA-II and NSGA-III.

In order to suppress the futile generation of excessive useless solutions the trade-off and distribution of a solution should be made an inherent solution property, based on which they can be sorted and selected. Solutions that are generated via NSGA-II and NSGA-III can be considered as units with multiple properties. This is, for the case of NSGA-II, graphically represented in Figure 6.1:

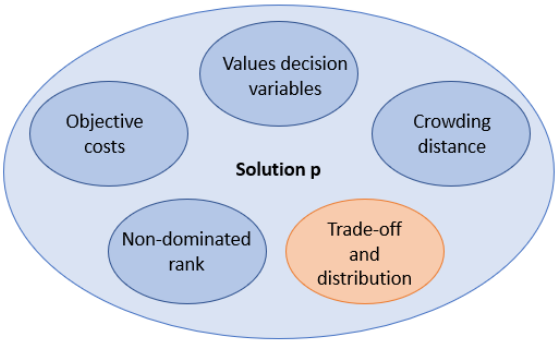


Figure 6.1: Graphical representation of a solution p, generated via NSGA-II, with its inherent solution properties (blue ovals) and its trade-off and distribution as an additional solution property (orange oval).

6.2.1 PIT-region

The trade-off Δt and distribution Δr of a solution p can be seen as properties imposed on the spatial distribution of other solutions around solution p, and show therefore great resemblances to the crowding distance of a solution, as used in NSGA-II. The crowding distance of a solution is defined as the average length of the edges of the *cuboid* formed around the concerned solution, with its neighbouring solutions as vertices. Because the desired solution trade-off Δt and distribution Δr are generally different for each objective function, the spatial form that they describe will often be much more complicated than a cuboid. Mattson et al. (2014) presented in this respect the so-called region of practically insignificant trade-off, or the PIT-region. The PIT-region is graphically displayed in Figure 6.2 for a bi-objective optimisation problem. Solutions that are located in the PIT-region of a solution p (red solutions) have no significant trade-off in regard to solution p and also to do not contribute to the overall solution diversity. They render no added value to the solution population and can therefore be discarded. The solutions located outside of the PIT-region of the concerned solution p are kept in the population (green solutions).

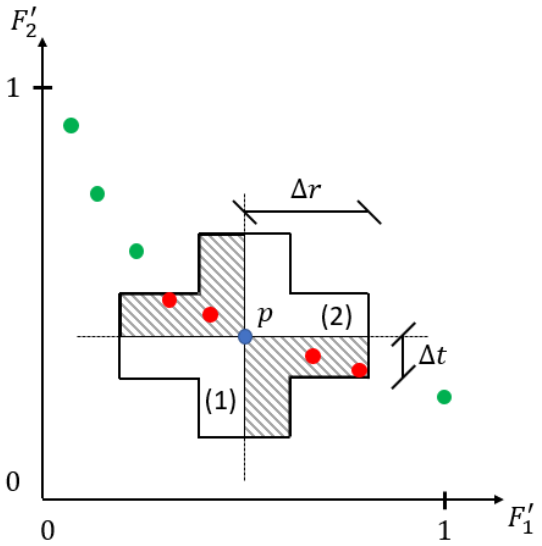


Figure 6.2: Graphical representation of the PIT-region of a solution p for an optimisation problem with normalised objectives F_1' and F_2' (adapted from Mattson et al. (2014)).

The PIT-region, as presented by Mattson et al. (2014), has a cross-like shape in case of a bi-objective problem. The PIT-region presented by Mattson et al. (2014) only consisted out of the shaded sections. Because in the presented trade-off function, the trade-off and distribution of solutions will be determined one non-dominated front at a time, the PIT-region can be simplified to a complete cross, built up out of rectangles circumventing the concerned solutions. Solutions that are located in sections (1) or (2) of the PIT-region will namely be part of a lower or higher non-dominated front respectively. Note that the dimensions of the PIT-region, defined by Δt and Δr, are not necessarily the same for each objective.

Mattson et al. (2014) used the PIT-region to reduce the size of a large solution population to eventually obtain two population sets: one set P_1 that contains all the discarded solutions and another set P_2 consisting out of the selected solutions that meet the required trade-off Δt and distribution Δr. In this scenario, all the solutions are sorted in ascending order according to the objectives. Subsequently the solutions are visited in ascending order and the PIT-region is constructed around the concerned solution. While the visited solution is allocated to the population set P_2, all the solutions that are located in its PIT-region are discarded and are allocated to population set P_1. The discarded solutions are not visited and their PIT-region is therefore not constructed.

6.2.2 Trade-off function

The presented trade-off function will not discard solutions based on their trade-off and distribution but will downgrade them. Just like solutions with a low crowding distance are disfavoured and downgraded during the selection procedure, solutions with a crowded PIT-region are disfavoured as well. The main aim of the trade-off function is to count the number of solutions in the PIT-region of a concerned solution. A so-called trade-off counter is added as an additional solution property. The more crowded the PIT-region of a solution is, the higher its trade-off counter will be. As a result of the cross-like shape of the PIT-region, solutions that are located in high trade-off areas of the Pareto front, i.e. steep segments and knees, will have a sparsely crowded PIT-region and their trade-off counter will therefore amount to a low value. Solutions located in the flat, low trade-off areas of the Pareto front on the other hand, will have a densely crowded PIT-region and their trade-off counter will amount to a high value. The pseudo code of the trade-off function is presented in Algorithm 5.

Algorithm 5 Trade-off function

Require: pop, $\mathcal{F}$, Δt, Δr
  for i=1 **to** $|pop|$ **do**
    % Reset trade-off counter
    $pop(i)$.TradeOffCounter=0
  end for
  $pop_n = \texttt{NormaliseCosts}(pop)$
  for i=1 **to** $|\mathcal{F}|$ **do**
    $pop_i = \texttt{SelectSolutions}(pop_n, \mathcal{F}_i)$
    $\texttt{Costs} = \texttt{SelectCost}(pop_i)$
    for j=1 **to** M **do**
      Costs=SortCosts(pop_i.Cost(j), ascending)
      for k=1 **to** |Costs| **do**
        itUp=1
        itDown=1
        while k-iterDown>0 **do**
          if InPIT(Costs(itDown), Costs(k), Δt, Δr) **then**
            $pop(k)$.TradeOffCounter++
          end if
          itDown++
        end while
        while k+itUp<|Costs| **do**
          if InPIT(Costs(itUp), Costs(k), Δt, Δr) **then**
            $pop(k)$.TradeOffCounter++
          end if
          itUp++
        end while
      end for
    end for
  end for

The trade-off function requires a solution population pop, which is sorted in its non-dominated fronts $\mathcal{F}$, and the desired trade-off Δt and distribution parameter Δr which are pre-defined by the decision maker. For the sake of clarity it is assumed that Δt and Δr have the same value for all the M objective functions. In a first instance the trade-off counters of all the solutions in pop are reset to zero. Additionally, while Δt and Δr are disclosed as percentages, the objective costs of all the solutions in pop are normalised. This is done via Equation (3.7).

The trade-off counter of the solutions is determined at a rate of one non-dominated front at a time. A new solution population pop_i is constructed for each non-dominated front, which contains all the solutions that are allocated to this front. The objective costs of the solutions in pop_i are clustered in the $M \times |pop_i|$ objective costs matrix, Costs. One column of Costs contains all the objective costs of one solution and can be considered as column vector in the objective space between the objective space origin and the solution. The trade-off function continues by sorting the column vectors of Costs in ascending order according to the concerned objective of the optimisation problem. Subsequently, for each k-th column vector of the sorted objective cost matrix Costs, it is determined if the neighbouring objective costs, both in ascending en descending order according to the concerned objective, are located in its PIT-region. If this is the case

the trade-off counter of the solution corresponding with the objective costs in the k-th column vector, is incremented by 1. The developed Matlab code of the trade-off function `CalcTradeOff` is included in Appendix A.4.

The trade-off counter is subsequently used as an additional sorting parameter. Ahead of sorting the solutions of the combined solution set R_t according to their crowding distance and non-dominated rank, they are sorted according to their trade-off counter in ascending order. This result in emphasising solutions with a sparsely crowded PIT-region. Because the dimension of the PIT-region of a solution is not only dependent on a desired objective trade-off but also on a minimal solutions diversity requirement, solution diversity will be more emphasised in t-algorithms.

6.3 MID as a stopping criterion

At the end of the main loops of the NSGA-II and NSGA-III algorithms, as presented by Deb et al. (2002) and Deb and Jain (2014), a generation counter is incremented with one (see Figure 3.2 and Figure 3.7). This gives rise to the conclusion that the number of main loop iterations acts as the termination parameter of both algorithms. The authors, Deb et al. (2002) and Deb and Jain (2014), sometimes mention that a *maximum* number of iterations is defined and not an absolute number of iterations. Accordingly, one can be tended to conclude that the algorithm can be aborted prematurely if wanted before the pre-defined maximum number of iterations is reached. However, in the outlines of NSGA-II and NSGA-III there is no attention given to the calculation or generation of a different termination parameter which could override the default termination parameter (i.e. the number of iterations). Additionally, in regard of the case studies presented in Deb et al. (2002) and Deb and Jain (2014), only the Pareto-optimal solution set, achieved after a pre-defined number of iterations is presented. Although Deb et al. (2002) and Deb and Jain (2014) touch upon the implementation of a different and more advanced stopping criteria, this is not further elaborated.

The previous chapter undeniably showed that the current stopping criterion of both algorithms is flawed. As of the solution population entirely consist out of non-dominated solutions, but the iteration process is continued nonetheless, no improvement in solution convergence or diversity is observed, let alone the large amount of computation time that is unnecessary spend. In several case studies even a decline in convergence was observed. It is obvious that repeating the algorithm's main loop for a pre-defined number of times has no relation to the concerned optimisation problem. On the other hand, because no information is generated about the convergence or diversity of the solutions, the algorithms are not capable to determine whether continuing the iteration process is still beneficial or not.

The implementation of the convergence and solution diversity parameters MID and SNDS in the Matlab code gave rise to the following understanding: if a solution population has converged to the Pareto front of a constraint optimisation problem, the MID of the population will stabilise. While NSGA-II and NSGA-III are both elitist algorithms, once the solutions have converged to the Pareto front, continuing the iteration process will namely only result in the solutions fluttering on the Pareto front itself. Because the mean Euclidean distance between the solutions on the Pareto front and the Utopia point will not change as a result of this, the stabilising of the MID can be seen as signal that further iterating has become futile.

After each iteration of the main loop, the MID of the solution population is calculated and saved. If the solution population completely consists out of non-dominated solutions and it is found that the difference in MID between two consecutive generations drops below a pre-defined value, the iteration process is aborted prematurely. The pseudo code of this algorithm termination process is presented in Algorithm 6:

Algorithm 6 MID-based termination process

for i=1 **to** $MaxIt$ **do**
  $pop_i, \mathcal{F}_i = \texttt{mainLoop}(pop_{i-1}, p_c, p_m, \sigma, \mu, \Delta t, \Delta r$,Obj, Constr$)$
  $MID_i = \texttt{CalculateMID}(pop_i)$
  $FPOS_i = |\mathcal{F}_{i,1}|/N$
  if $|MID_i - MID_{i-1}| \leq \Delta MID$ **and** $FPOS_i = 1$ **then**
    breakMainLoop
  end if
end for

It is still necessary for the decision maker to pre-define a maximum number of iterations $MaxIt$. After the main loop is completed, the MID and $FPOS$ of the generated solution population pop_i are calculated. Then it is determined if the difference between the MID-value of the current population and the previous one, is smaller than a pre-defined ΔMID value. This ΔMID represents the maximum acceptable deviation in MID to consider the two values as equal. If the difference between the MID-values of two consecutive populations is indeed smaller than ΔMID and $FPOS = 1$, the main loop is stopped.

6.4 The tM-evolutionary algorithm

The basic framework of the tM-algorithms is the analogous of the constrained algorithms. The pseudo codes of tM-NSGA-II and tM-NSGA-III are included in Appendices C.1 and C.2 respectively. The four major changes are:

1. The initial solution population contains the anchor points of the M objectives.
2. The trade-off function is incorporated, following the non-dominated sorting functions.
3. The function used to sort and select N solutions from the combined solution set is replaced by the $\texttt{t}-\texttt{SortPopulation}$.
4. The MID of the solution population is used as a stopping criterion.

The anchor points, or individual minimisers, of the objective functions are determined with the built-in fmincon function in Matlab. While the tM-NSGA-II and tM-NSGA-III algorithms are stochastic, the fmincon Matlab function uses a deterministic algorithm. The anchor point of an objective is determined via a single objective optimisation problem. The M anchor points are included in the initial solution population to ascertain the extremities of the Pareto front are sufficiently explored, which often was a problem in the constrained algorithms. The often unsatisfactory solution diversity of the constrained algorithms can namely be the result of an initial population with an insufficient solution diversity. Because the anchor points are non-dominated and contribute to the solution diversity, they will be retained in the subsequent solution populations.

The initial idea was to use the anchor points to create the CHIM in the objective space (see 2.3.2). The formula which mathematically describes the CHIM could be used as an additional constraint for the initial population. This would allow to create a uniformly spread initial population, located on the CHIM. This would simultaneously increase the convergence speed and solution diversity of the final solution population. In order to create solutions located on the CHIM, a deterministic algorithm would be needed. This would however bypass the need for an evolutionary algorithm, while the deterministic algorithm can be as easily used to generate solution on the Pareto front rather than solutions on the CHIM. Only including the anchor points in the initial population does not bypass the need for an evolutionary algorithm and therefore

the CHIM is not constructed.

Note that the remaining $N - M$ initial solution are randomly generated in the feasible space. Nonetheless, it could be beneficial to structure the feasible space. This could be done with a grid for instance. Then, instead of generating random solutions in the feasible space, the vertices of the grid can be used as the initial solution population. If these vertices are located outside of the feasible space, a random solution can be generated within the feasible space instead. This would eventually result in a more structured initial population with a high solution diversity, containing both structured and random solutions. Figure 6.3 graphically displays this process for a bi-objective case study. The structured initial population can have the same benefits as an initial population located on the CHIM, but does not require the use of deterministic algorithms. This structured initial solution population is not implemented in the tM- or the tDOM-algorithms but is only cited as a possible additional improvement.

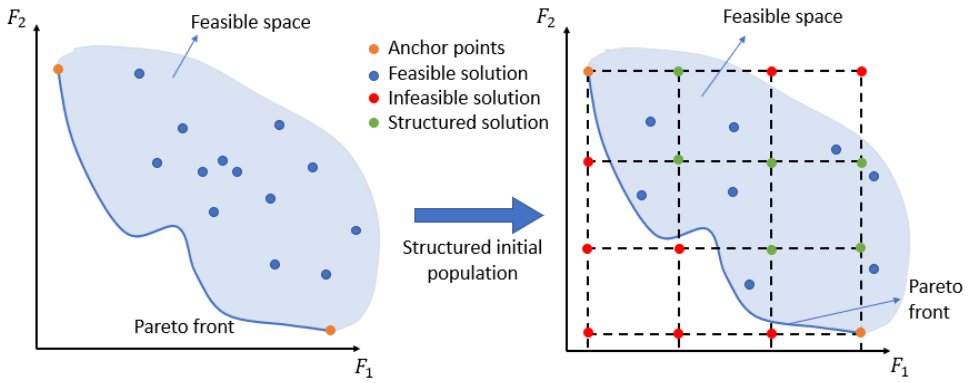


Figure 6.3: Example of a possible scheme to structure the initial population.

One of the two main improvements of the tM-algorithm, is the implementation of the trade-off function. While the trade-off function can be seen as an additional crowding distance, it is incorporated in the algorithm after the calculation of the crowding distance (in the case of tM-NSGA-II), or the allocation of solutions to the reference points (in the case of tM-NSGA-III). Because the solution population has to be already sorted in its non-dominated fronts, the trade-off function must always be preceded by the non-dominated sorting function.

The sorting function is extended with a sorting step based on the trade-off of the solutions, rendering the t-sorting function. The function consists out of three subsequent sorting steps. In a first step, the solutions of the population are sorted in ascending order, based on the value of their trade-off counters. The sorting than continues with the two sorting steps that are also included in the constrained NSGA-II and NSGA-III algorithms. Firstly, the solutions are sorted based on their crowding distance (in the case of NSGA-II) or distance to their associated reference point (in the case of NSGA-III) in descending or ascending order respectively. To conclude, the solutions are sorted in ascending order based on their non-dominated rank. As a result of the three sorting steps of the t-sorting function, the solutions with a sparsely populated PIT-region, high contribution to the overall solution diversity, and low non-dominated rank will be sorted in the highest positions of the population.

The novel stopping criterion is incorporated as the last step of the tM-algorithms' main loop. Both the MID and FPOS of the generated population are calculated. If the difference in MID of two consecutive solution populations is found to be lower than a user pre-defined ΔMID, the algorithm's main loop is stopped. The algorithm, and Matlab code, are however thus programmed that the algorithms can only be stopped

if, additionally, the FPOS of the current population has reached it's maximum value of 1. This is done to avoid the algorithm from stopping if the amount of non-dominated solutions is not sufficient yet. The combination of both the FPOS and MID renders a stopping criterion which has an increased relevance to the concerned optimisation problem than the default stopping criterion of reaching a pre-defined number of iterations.

An additional modification that is made in the tM-algorithms, is the abolishment in futile use of the time consuming non-dominated sorting function and diversity-enabling functions. If, for instance, the constrained NSGA-II algorithm (see B.1) and tM-NSGA-II algorithm (see C.1) are compared, this cutback can be clearly seen. During the initialisation step, the non-dominated sorting function, crowding distance calculating step, and sorting step are discarded. This is justified by the fact that the initial population is only needed to create offspring solutions via a mutation and crossover step. The obtained offspring solutions set is subsequently merged with the parent solutions set. It is only sensible to determine the non-dominated ranks and crowding distances of the solutions in this combined solution population while this population has to be sorted in order to select the best N solutions. After the selecting procedure, the constrained NSGA-II algorithm continues by updating the non-dominated ranks and crowding distances of the selected solutions. From a practical point of view, it is however only sensible to update the non-dominated ranks of the selected solutions. Therefore, in tM-NSGA-II, the crowding distances update is discarded.

The major expectations for the tM-algorithms' performance and solution quality are the following:

1. A significant decrease in computation time.
2. A higher solution diversity.
3. A Pareto front with a high solution density in high trade-off areas and a low solution density in low trade-off areas.

6.5 Case studies

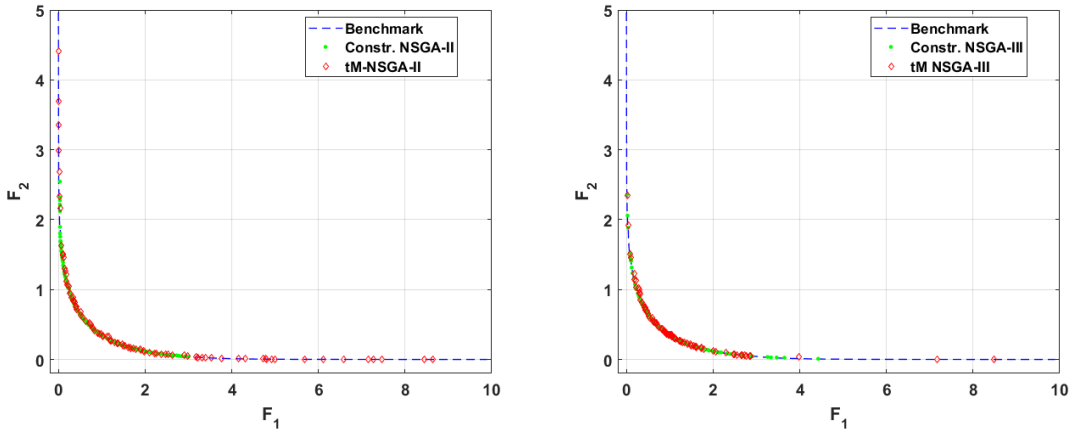
tM-NSGA-II and tM-NSGA-III are tested on the case studies that are presented in 4.3. The many-objective case study is not further examined, based on the reasoning quoted in 5.3.3. The used settings for the EA parameters, used for each case study, are summarised in Table 6.1.

Table 6.1: The EA parameters settings, used for each case study.

Parameter	Value
Maximum iterations	75
Population size	100
Crossover probability p_c	90.0 %
Mutation probability p_m	10.0 %
Mutation rate μ	5.00 %
Mutation step size σ	$0.05 \times (\mathbf{b} - \mathbf{a})$
Objective subdivisions	99 (*), 13 (**)
Reference points	100 (*), 105 (**)
Trade off Δt	5.00 %
Distribution parameter Δr	10.0 %
ΔMID	1.00 %

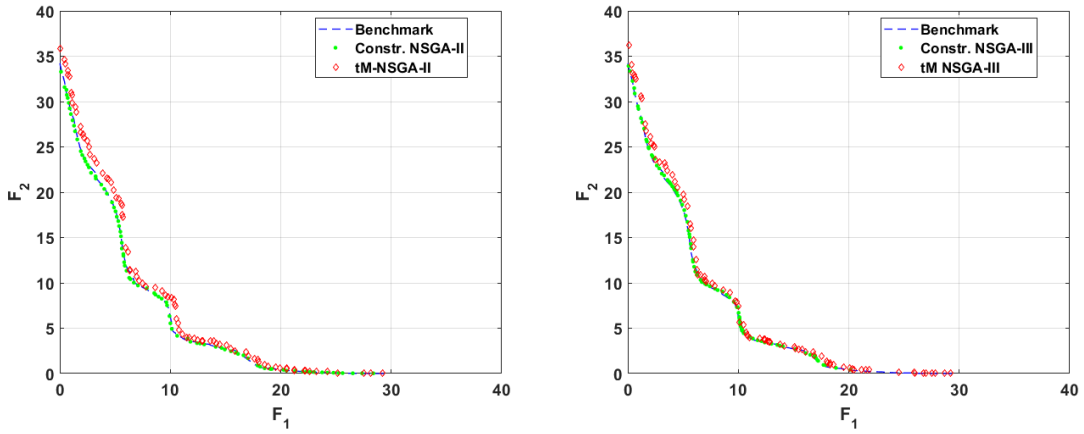
Just like for the constrained NSGA-II and NSGA-III algorithms, the number of objective subdivisions d in the tM-NSGA-III algorithm is set so that the number of reference points $H \geq N$. The trade-off Δt and distribution parameter Δr have the same value for each objective, for the sake of clarity.

6.5.1 The bi-objective case studies



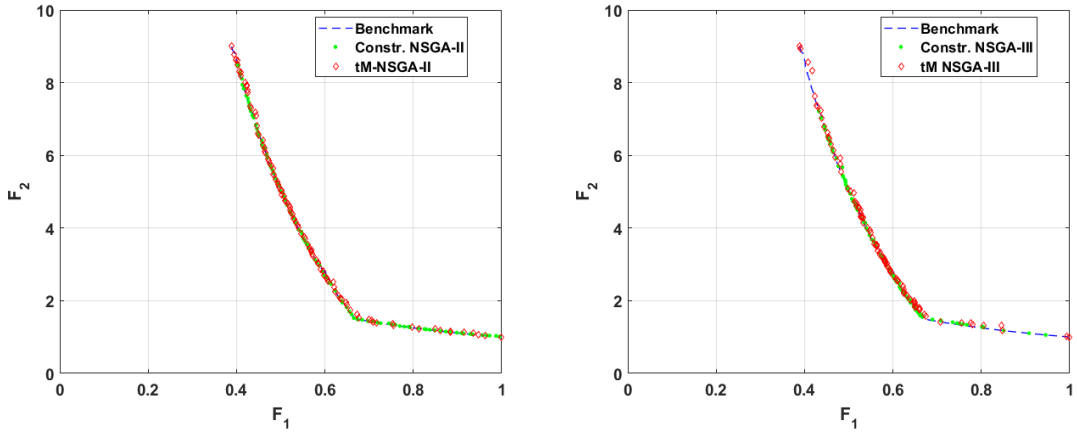
(a) tM-NSGA-II **(b)** tM-NSGA-III

Figure 6.4: Pareto front of the BIOBJ-problem generated via tM-NSGA-II (a) and tM-NSGA-III (b).



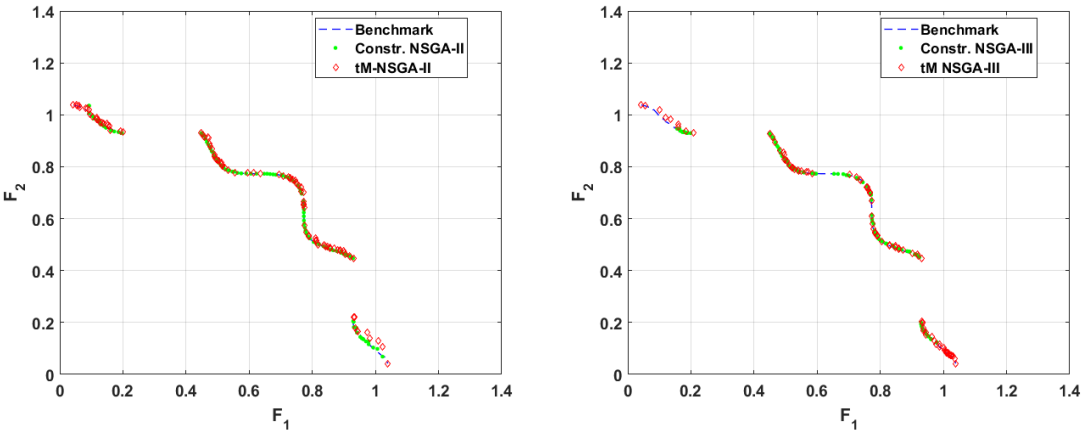
(a) tM-NSGA-II **(b)** tM-NSGA-III

Figure 6.5: Pareto front of the DO2DK-problem generated via tM-NSGA-II (a) and tM-NSGA-III (b).



(a) tM-NSGA-II **(b)** tM-NSGA-III

Figure 6.6: Pareto front of the CONSTR-problem generated via tM-NSGA-II (a) and tM-NSGA-III (b).



(a) tM-NSGA-II **(b)** tM-NSGA-III

Figure 6.7: Pareto front of the TNK-problem generated via tM-NSGA-II (a) and tM-NSGA-III (b).

Based on the graphical representation of the Pareto fronts of the bi-objective case studies, generated via tM-NSGA-II and tM-NSGA-III, already one improvement can be clearly seen, namely the increased spread of solutions on the Pareto front. Especially the BIOBJ-problem displayed an unsatisfactory solution diversity in case of the constrained algorithms. Now however, solutions are generated over the complete length of the Pareto front and not only in the vicinity of the local high trade-off area (see Figure 6.4). For the ease of formulation, solutions generated via the tM-algorithms will be referred to as ”tM-solutions”, and solutions generated via the previously discussed constrained algorithms will be referred to as ”constrained-solutions”.

Another difference between the constrained- and tM-solutions, is that the latter show a variable resolution, based on the trade-off of the Pareto front area in which they are located. In the instance of a bi-objective optimisation problem, Pareto front areas that are parallel with an objective especially display a low trade-off. In all the Pareto fronts of the bi-objective case studies, it can be seen that the tM-solutions are more sparse in these areas than the constrained-solutions. The high trade-off areas of the Pareto front how-

ever, show a much higher tM-solution density. Incorporating this trade-off based Pareto front resolution in evolutionary algorithms, was one of the main goals of this thesis. The fact that the filtering of solutions based on their trade-off, is obtained without the generation of an excess amount of solutions, is most certainly an advantage.

Concerning the DO2DK-problem (see Figure 6.5), it can be seen that for both tM-algorithms, the constrained solutions display a better convergence than the tM-solutions. This is especially the case for the tM-NSGA-II algorithm. A similar problem can be established for the TNK-problem (see Figure 6.7). In that case, the tM-NSGA-II algorithm displays a bad convergence at the extremities of the Pareto front. Generally, the tM-NSGA-III algorithm seems to display a better convergence, however the solution diversity seems to be higher for the tM-NSGA-II algorithm. In comparison with the constrained algorithms, both tM-algorithms display a higher solution diversity and a trade-off based Pareto front resolution, but a slightly worse convergence.

In order to objectively compare the tM-algorithms with each other and with the constrained algorithms, the mean performance parameters and runtime of the two tM-algorithms are summarised in Table 6.2. The last column of Table 6.2 contains the relative difference between the best value of the performance parameter or runtime of the tM-algorithm, and the corresponding best value found for the constrained algorithm.

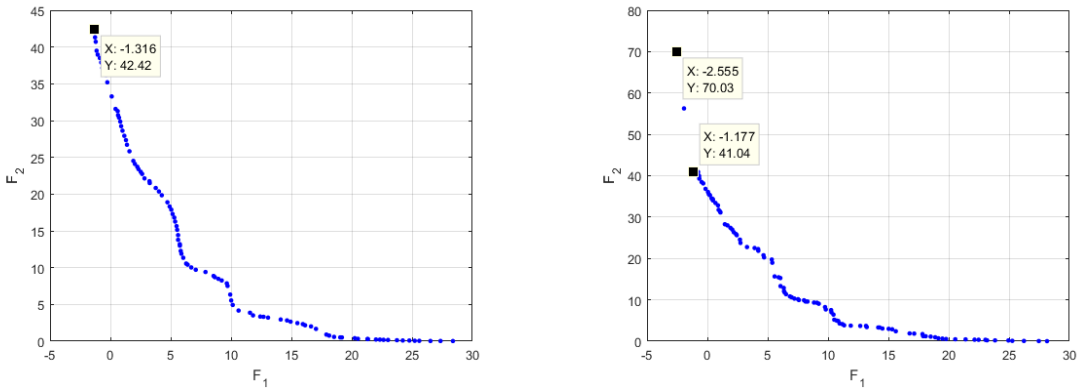
Table 6.2: Performance parameters and runtime of the bi-objective case studies for the tM-algorithms. The best value, per case study, of the concerned performance parameter is set in bold. The last column contains the ratio of the best value of the tM-algorithms to the best value of the constrained algorithms.

	tM-NSGA-II		**tM-NSGA-III**		**Constr. EA**
Performance parameter	**Mean**	**St. dev.**	**Mean**	**St. dev.**	**Ratio tM/Constr. [%]**
BIOBJ-problem					
Iterations [-]	24.1	1.6	**23.5**	**1.4**	-68.67
MID [-]	0.2780	0.0350	**0.2308**	**0.0213**	-47.27
SNDS [-]	**0.1772**	**0.0277**	0.1518	0.0241	-4.78
Runtime [s]	**74.3129**	**5.1056**	110.8429	19.5521	-65.90
DO2DK-problem					
Iterations [-]	**4.4**	**0.7**	4.9	1.3	-94.13
MID [-]	0.5392	0.0147	**0.5236**	**0.0166**	-15.96
SNDS [-]	**0.1590**	**0.0072**	0.1566	0.0140	-3.81
Runtime [s]	**23.0996**	**3.2567**	30.9766	6.5203	-90.11
CONSTR-problem					
Iterations [-]	14.5	1.1	**13.4**	**1.6**	-82.13
MID [-]	0.5811	0.0263	**0.5403**	**0.0378**	-6.76
SNDS [-]	**0.1718**	**0.0047**	0.1693	0.0060	-0.69
Runtime [s]	**51.3015**	**3.7196**	61.4631	7.7600	-74.43
TNK-problem					
Iterations [-]	**13.9**	**1.0**	14.8	1.6	-81.47
MID [-]	0.9453	0.0030	**0.9442**	**0.0053**	+0.26
SNDS [-]	0.3134	0.0154	**0.3197**	**0.0155**	+6.50
Runtime [s]	**50.9809**	**4.4607**	69.3380	7.2593	-67.34

In Table 6.2, the FPOS value is not included because this value is defined by the tM-algorithms' structure and always equals 1.

Table 6.2 confirms the assumption made based on Figures 6.4 to 6.7, that the tM-NSGA-III algorithm display the best convergence. A surprise however, following from Table 6.2 is that in all the case studies (except for the CONSTR-problem), the MID has decreased. This decrease could be the result of an increased solution convergence. However this is found to be unlikely, mainly based on the graphical representations of the Pareto fronts in Figures 6.4 to 6.7. For instance, the DO2DK-problem displays both for tM-NSGA-II and tM-NSGA-III a decreased solution convergence. Nonetheless, based on Table 6.2, the best average MID value of the tM-algorithms is 15.96 % smaller than the best average MID value of the constrained algorithms. This decrease in MID should suggest an increased solution convergence, but from a visual comparison this is found not to be the case.

The decrease in MID is most likely the result of the incorporation of the anchor points in the initial population. As already mentioned, the anchor points are likely to remain in the subsequently generated solution populations up to and including the final solution population. Therefore, the objective costs of the anchor points provide, on one side, the Utopia point, but also the Nadir point of the population. Before calculating the MID of the final solution population, the objective costs of the solutions in the population are normalised. This is done based on Equation (3.7), which requires the Utopia and Nadir point of the population. If however the anchor points remain remote of the bulk of the solutions in the population, the normalisation based on Equation (3.7) will be disproportionate, and eventually rendering an unjustified decrease in MID. Figure 6.8 displays the Pareto fronts of the DO2DK-problem, only this time with objective axes that also contain negative objective values. In all the other representations of Pareto fronts thus far, the objective axes were cropped to only contain positive objective values, whilst negative objective values are in most cases practically irrelevant.



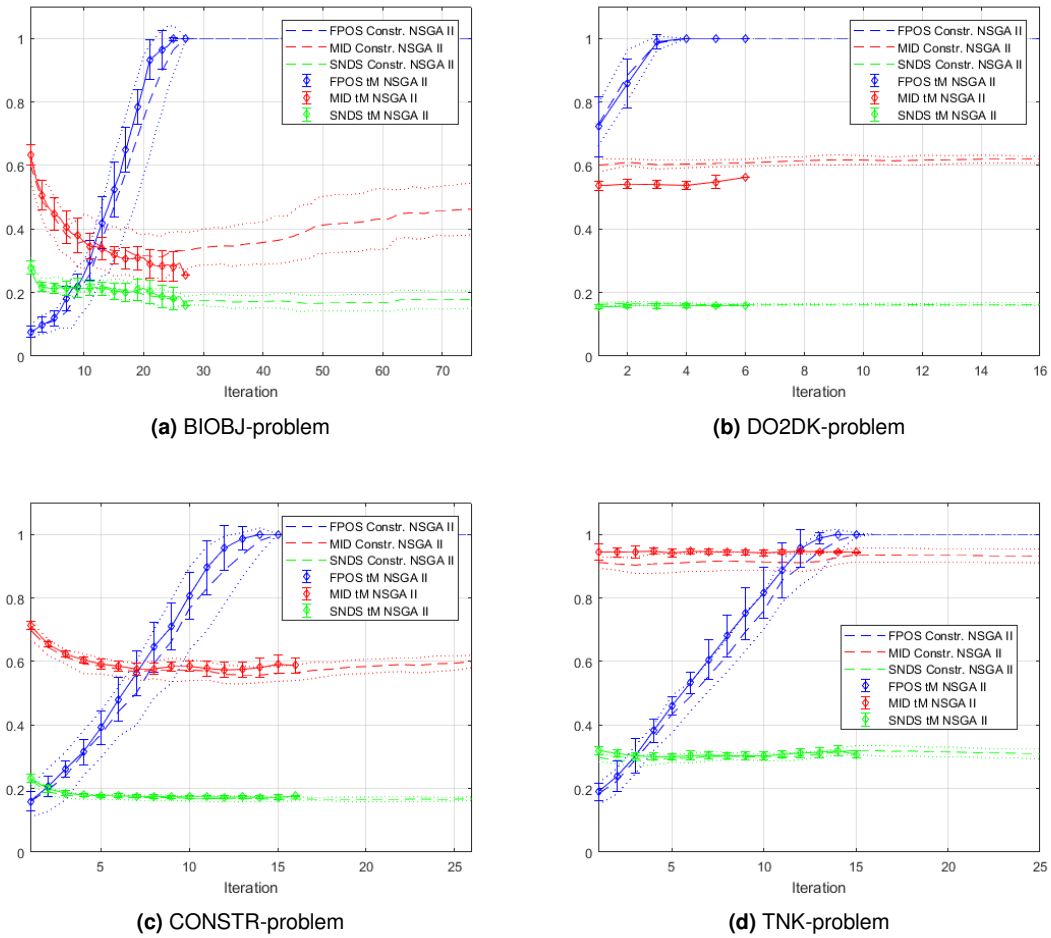
(a) Constrained NSGA-II **(b)** tM-NSGA-II

Figure 6.8: Pareto fronts of the DO2DK-problem with negative objective axes, generated via constrained NSGA-II (a) and tM-NSGA-II (b).

Figure 6.8(b) clearly shows that the anchor point of the first objective is remote from the bulk of the generated solutions. In comparison with Figure 6.8(a) it can be seen that, except for two solutions, all the tM-solutions are located within the same range as the constrained-solutions. In Figure 6.8 this is indicated by the cost tags. Because the remote anchor point of the first objective is however used to normalise the solution population, the range within the solutions are located, becomes immediately significantly larger in case of the tM-NSGA-II algorithm. The normalised Euclidean distance between the majority of the tM-solutions and the Utopia point will therefore decrease, be it artificially, in comparison with that of the constrained-solutions. Because the bulk of the solutions are located within the same objective range, the standard deviation of these normalised Euclidean distances will remain roughly the same. Table 6.2 con-

firms this. If the two remote tM-solutions are left out of the solutions population, the MID of the tM-solutions amounts to 0.6315, which is, as expected based on Figure 6.5, 1.35 % higher than the best average MID of the constrained algorithms for the DO2DK-problem. Note that in case of the BIOBJ-problem, the same phenomenon occurs.

Continuing with the discussion of the results summarised in Table 6.2, the most striking difference between the constrained-algorithms and the tM-algorithms, is the decrease in runtime and iterations that were needed. The novel stopping criterion, based on the MID and FPOS of the solution population, resulted in an average 74.45 % decrease in computation time. If the two tM-algorithms are mutually compared on the basis of their required runtime and iterations, it can be seen that tM-NSGA-II outperforms tM-NSGA-III in the case of all four bi-objective case studies in terms of runtime. However, in terms of needed iterations, tM-NSGA-III outperforms tM-NSGA-II in two cases. It would expected that if an algorithm requires less iterations, it would also require less runtime, but this is not the case. The only difference between the tM-NSGA-II and tM-NSGA-III algorithm is the use of reference points. Based on the results displayed in Table 6.2, the assumption made in the previous chapter (see 5.3.1), which states that the allocation of solutions to the reference points is cumbersome, is hereby proven.



(a) BIOBJ-problem

(b) DO2DK-problem

(c) CONSTR-problem

(d) TNK-problem

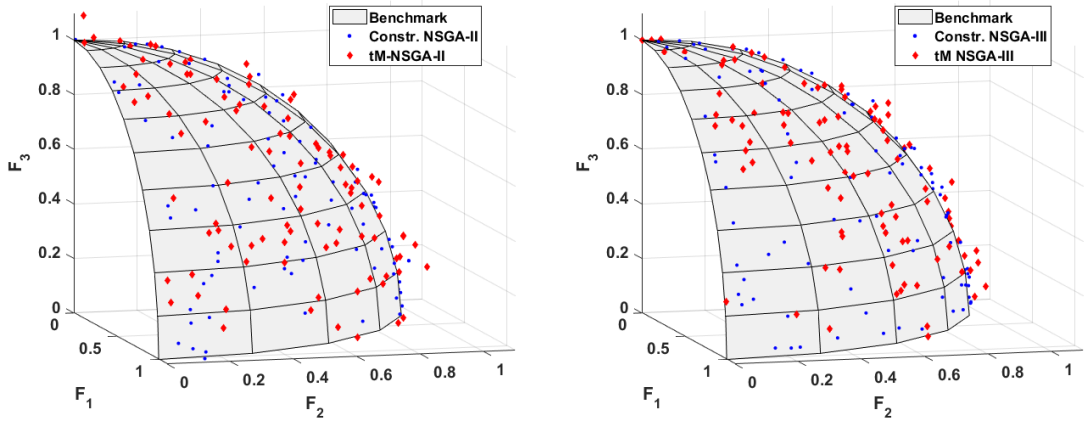
Figure 6.9: Performance plots of the tM-NSGA-II algorithm and constrained NSGA-II algorithm for the bi-objective case studies (average taken over 10 repetitions). The iteration axis is cropped.

Figure 6.9 graphically displays the performance plots of the tM-NSGA-II algorithm and the constrained

NSGA-II algorithm for the bi-objective case studies. For the sake of clarity, the iterations axis is cropped. The according performance plots for the tM-NSGA-III and constrained NSGA-III algorithm, are included in Appendix C.3. In Figure 6.9, the dashed line represents the average value of the concerned performance parameter for the constrained-solutions, whilst the dotted line represent the standard deviations.

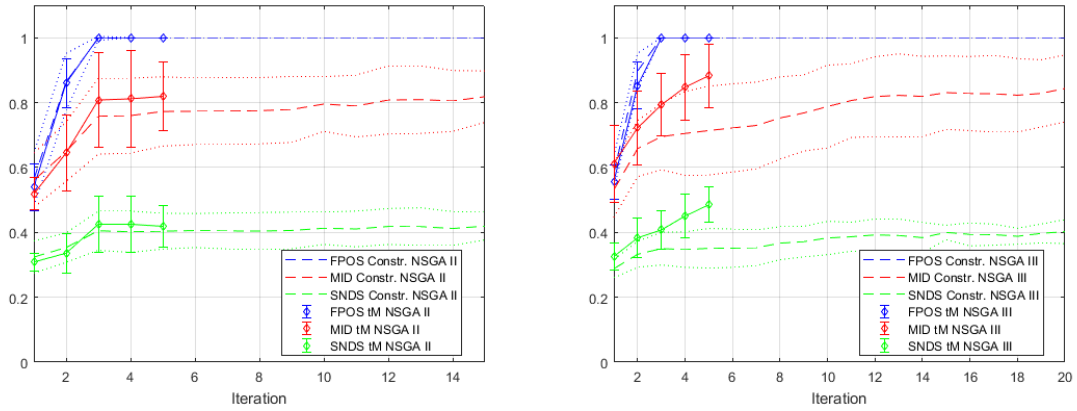
Regarding the progress of the FPOS performance parameter, it is apparent based on Figure 6.9, that the speed with which non-dominated solutions are generated, is slightly higher in case of the tM-algorithms. For the BIOBJ-, CONSTR-, and TNK-problem, the average FPOS of the tM-solutions during a particular iteration step, is higher than the average FPOS of the constrained-solutions during the same iteration step. The fraction non-dominated tM-solutions that have been generated up until that iteration step, is therefore higher than the fraction non-dominated constrained-solutions that had been generated thus far. In Figure C.1, this phenomenon is even more pronounced.

6.5.2 The three-objective case study



(a) tM-NSGA-II **(b)** tM-NSGA-III

Figure 6.10: Pareto front of the DTLZ2.3-problem, generated via tM-NSGA-II (a) and tM-NSGA-III (b).



(a) tM-NSGA-II **(b)** tM-NSGA-III

Figure 6.11: Performance plots of the tM-NSGA-II and tM-NSGA-III algorithms for the DTLZ2.3-problem (average taken over 10 repetitions). The iteration axis is cropped.

The Pareto fronts of the DTLZ2.3-problem that are generated by the tM-algorithms, display a high solution diversity and uniform solution spread. This however was also the case for the constrained-solutions, so at first glance no improvement in solution diversity is witnessed. Additionally it is noticeable, based on Figure 6.10 that the convergence of the tM-solutions is worse than that of constrained-solutions. This loss in convergence was also noticed in the bi-objective case studies. Based on a visual comparison, tM-NSGA-II displays the worst solution convergence. The performance parameters in Figure 6.11 and Table 6.3 confirm this assumption.

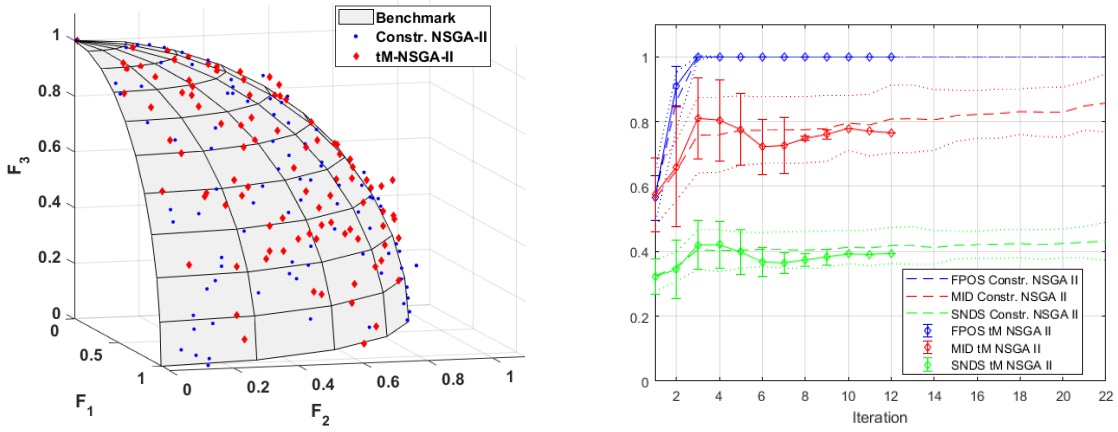
Although the MID of the tM-NSGA-III algorithm is higher than that of tM-NSGA-II algorithm, the former displays a better convergence. The increased MID of the tM-NSGA-III algorithm is again the result of an increased solution diversity. The SNDS namely displays a similar progress in function of the number of iterations like the MID. The tM-NSGA-II algorithm displays a similar SNDS and MID progress, yet less pronounced. In Table 6.3 a comparison is made between the best performance parameters of the tM-algorithms and the best corresponding performance parameters of the constrained-algorithms. Again, the most significant improvements are made in terms of the number of iterations that are needed and computation time.

Table 6.3: Performance parameters and runtime of the tM-algorithms for the DTLZ2.3-problem. The best value of the concerned performance parameter is set in bold.

	tM-NSGA-II		**tM-NSGA-III**		**Constr. EA**
Performance parameter	**Mean**	**St. dev.**	**Mean**	**St. dev.**	**Relative difference [%]**
Iterations [-]	**4.3**	**0.5**	4.4	0.5	-94.27
MID [-]	**0.8107**	**0.1395**	0.8472	0.0948	-10.73
SNDS [-]	0.4242	0.0794	**0.4511**	**0.0636**	-2.02
Runtime [s]	**22.1558**	**2.7660**	29.0646	3.9129	-89.81

Because the performance parameter indicating the convergence of the solutions is used as a stopping criterion, it might be so that if the stopping criterion ΔMID is set more rigorously, in this case to 0.50 %, the tM-solutions display a better convergence. This scenario is represented in Figure 6.12. By decreasing the allowed tolerance in convergence between two subsequent solution populations, the solutions indeed display an increased overall convergence. Figure 6.12(b) shows that the iteration process is continued for

a longer amount of iterations after the FPOS value has reached its maximum value. If the ΔMID value was set to 1.00 %, the tM-algorithm was aborted more quickly and, based on the bad solutions convergence, too hastily. This could also have been deduced based Figure 5.7, in which it is visible that the MID and SNDS of the DTLZ2.3-problem stabilise more in case of a higher number of iterations.



(a) Pareto front **(b)** Performance plot

Figure 6.12: Graphical representation the Pareto front and performance plot of the solutions generated with tM-NSGA-II and ΔMID = 0.50 %. The iteration axis of the performance plot is cropped.

6.5.3 General observations

The major recorded improvement of the tM-algorithms is undoubtedly the saving of computation time. In all the case studies, significantly less iterations were used to generate a solution population with the required properties. In case of the bi-objective case studies, the Pareto fronts clearly displayed a trade-off and distribution dependent resolution and this was achieved without the excess generation of solutions. Also the incorporation of anchor points rendered a more diverse solution population, and most likely additionally increased the speed at which non-dominated solutions were generated. The latter assumption is yet not proven. The three-objective case study did not show a significant increase in solution diversity, and because the high trade-off areas of the Pareto front are less distinct, the introduced trade-off based solution resolution was not clearly visible.

Both the bi-objective and the three-objective case studies however displayed a loss in solution convergence in comparison to the constrained-solutions. This can be undone by increasing the rigorousness of the ΔMID stopping criterion, but this is not practical. The major disadvantage of the tM-algorithms is that the used stopping criterion is still (be it arbitrary) defined by the user, just like the maximum number of iterations in the constrained-algorithms. Although the stopping criterion now has more relevance to concerned optimisation problem, it still requires an additional external parameter that has to be defined. The tDOM-algorithms, which are presented in the following chapter, have no need of an additionally externally defined stopping criterion, but instead use the density of solutions in the PIT-region as a stopping criterion.

The problem-dependent overall performance of the constrained-algorithms is not entirely resolved. The standard deviation of the performance parameters of the tM-algorithms, was still within the same range as those of the constrained-algorithms. However more consistency in the overall performance was observed. The tM-NSGA-III algorithm always displayed the best solution convergence for the bi-objective case studies, whilst the tM-NSGA-II was always the fastest and often rendered the most diverse solution

set. Also the convergence of constrained-solutions to local optima was not perceived for the tM-solutions. The question is however if this problem-dependent performance is a practical issue. Both tM-algorithms namely show satisfactory results on convergence, speed, and solution diversity.

6.6 Conclusion

In the previous chapter, the constrained-NSGA-II and NSGA-II algorithms were presented. These algorithms showed however several shortcomings. The constrained algorithms were time consuming, had a naive and problem irrelevant stopping criterion and the constrained-solutions tended to converge to local optima. The tM-algorithms, that were presented in this chapter, were developed to alleviate the shortcomings of the constrained algorithms on one side, but also to incorporate the trade-off of generated solutions as an additional selection criterion. The name ”tM-” is a contraction of trade-off and MID: the algorithm emphasises solutions with a high trade-off and uses the MID of the solution population as a stopping criterion.

In the first two sections of this chapter, the concepts of the trade-off function and MID-stopping criterion were further explained. The trade-off is incorporated in the tM-algorithms as a second crowding distance. To achieve this, the PIT-region, as presented by Mattson et al. (2014), is used to estimate the trade-off of a solution. However Mattson et al. (2014) uses this PIT-region to a posteriori reduce an initially large solution population to a small solution population that only contains solutions that meet the required solution properties. This method requires an excessive amount of solutions that have to be generated, only to be discarded during the filtering process. This means that a large amount of the computation time is used to generate futile solutions. Because time is money, Hashem et al. (2017) presented a ”divide-and-conquer” scheme for deterministic algorithms, that allows the algorithm to recursively generate solutions with a minimal trade-off without the excessive generation of futile solutions. The presented trade-off function also uses this idea, but adopted to evolutionary algorithms. A trade-off counter is introduced as an additional inherent solution property and the trade-off function is used to establish the amount of solutions in the PIT-region. The trade-off counter of the solution that is concerned, amounts to this number. The higher this value, the more crowded the PIT-region of that solution is, and the less favourable the solution is in terms of trade-off or attribution to the overall population diversity. Solutions with a high trade-off counter are therefore downgraded, just like those with a high non-dominated rank or low crowding distance.

The second main feature of the tM-algorithms, is the use of the MID as a stopping criterion. Because NSGA-II and NSGA-III are elitist algorithms, solutions will only move about on the Pareto front itself once they have converged and the iteration process is not stopped. This movement of solutions on the Pareto front however does not change the overall convergence, or MID, of the population. Thus the stabilisation of the MID is a signal that the solution population has converged to the Pareto front and the algorithm can be stopped.

A last additional feature of the tM-algorithms, is the incorporation of the individual minimisers, or anchor points, of the objectives in the initial population. The anticipation is that this will allow for a diverse initial population, easing the generation of diverse and non-dominated solutions. The idea of structured initial population is touched upon, but is not incorporated in the tM-algorithms.

The general expectations of the tM-algorithms’ performances for the case studies are a significant decrease in computation time, the generation of a Pareto front with a trade-off based resolution, and a higher solution diversity. The first expectation, i.e. the decrease in computation time, was satisfied for all the case studies. The tM-algorithms also generated, especially in the case of the bi-objective case studies, Pareto fronts with a high solution density in high trade-off areas and a low solution density in the other

Pareto front areas. However, and although the MID performance parameter indicated differently due to a disproportionate normalisation procedure, the convergence fo the tM-solutions was worse than that of the constrained-solutions. Additionally it was found that the value of the user-defined ΔMID had an influence on this problem. The more rigorous the MID-stopping criterion was set, the better the convergence.

In spite of the fact that the MID-stopping criterion is one with relevance to the concerned problem, its value still has to be pre-defined by the user. Because the user only has well founded requirements in terms of minimal solution trade-off and distribution, the MID-stopping criterion is anew an arbitrary defined parameter. The user namely has no founded estimation of what a good solution convergence is for the concerned optimisation problem, and thus how much tolerance, or ΔMID, that can be allowed. Stimulated by this flaw, the algorithms presented in the next chapter will use the trade-off and distributions of solutions as a stopping criterion. These algorithms are the so-called tDOM-algorithms.

Chapter 7

tDOM-NSGA-II and tDOM-NSGA-III

7.1 Introduction

The major shortcomings of the constrained-algorithms were largely resolved via the tM-algorithms that were presented in the previous chapter. The computation time was significantly reduced, the Pareto front had a trade-off based solution resolution and the solution diversity was improved (in the sense that the tM-solutions no longer converged to local optima). However, one major shortcoming was found: the use of the difference in MID, or convergence, between two subsequent solution populations is no valid stopping criterion. Although the MID-stopping criterion had more relevance to the concerned optimisation problem than the default iteration-stopping criterion, it was still an additional parameter which had to be pre-defined by the user. Because the user only has founded requirements for the number of solutions that have to be generated, and the minimal trade-off and distribution they must display, the MID-stopping criterion was anew an arbitrary stopping criterion. The user namely has no idea what a good solution convergence is for the concerned optimisation problem, and therefore also has no idea what tolerance, or ΔMID, that can be allowed.

Moreover it can be stated that the MID-based stopping criterion is an unnecessary complication. Because the tM-algorithms were capable of determining the value of the solutions it generated, in terms of trade-off and distribution, it also is capable to determine if the continuation of the iteration process is still meaningful. If it is found that this is not the case, the algorithm itself should be able to generate a termination parameter which can overrule the pre-defined number of iterations. This would mean that the discontinuation (or continuation for that matter) of the iteration process is no longer dependent on an externally and arbitrary defined parameter, but on inherent, problem related parameters which are formulated by the user based on underpinned considerations.

Hashem et al. (2017) also presented this idea. In their ”divide and conquer” scheme for deterministic algorithms, the exploration of a certain area of the Pareto front is ceased if the generated solution no longer displays the required trade-off or distribution. The so-called tDOM-algorithm, which are presented in the remainder of this chapter, adapt the concept stated above to evolutionary algorithms. This is also achieved by the use of the PIT-region. The name ”tDOM-algorithm” is a reference to the algorithms application of t-domination.

In the remainder of this chapter, the adapted trade-off function will be presented. This trade-off function will allow the algorithm to determine the trade-off counter of the generated solutions, but also the degree in which they are different to the solutions of the previously generated population. The latter property will

be determined via the use of t-domination, which is a variant non-domination. t-domination also takes the solution trade-off into account, besides its objective costs. After the introduction of the adapted trade-off function, the complete tDOM-algorithms will presented and will be subsequently tested on the numerical case studies that were presented in 4.3.

7.2 Trade-off as a stopping criterion

In the previous chapter, the concept of the PIT-region (see 6.2.1) and trade-off function were introduced (see 6.2.2). The trade-off function at hand however only used the minimal required trade-off Δt and distribution Δr as an additional crowding distance or diversification measure. The trade-off function that will be presented in this chapter, will also evaluate if the algorithm can be stopped. In order to do so, two additional inherent solution properties are added. For the ease of formulation, solutions that are generated via a tDOM-algorithm, will be further denoted as tDOM-solutions.

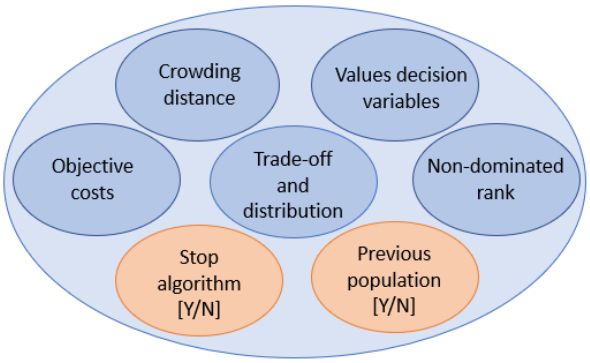
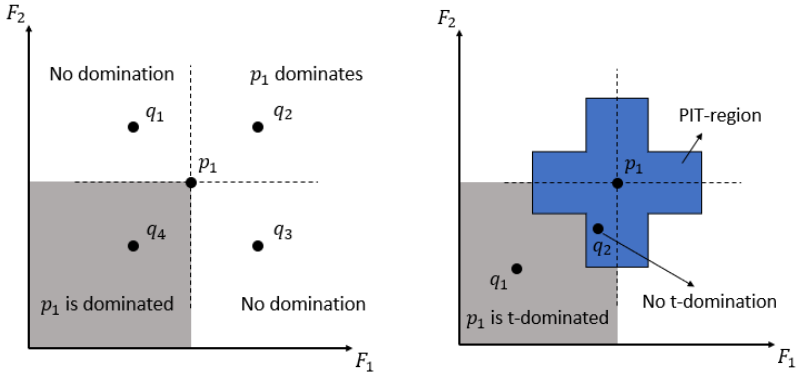


Figure 7.1: Graphical representation of a tDOM-solution with its inherent solution properties.

Figure 7.1 graphically displays a tDOM-solutions with its inherent solution properties. The two added solution properties are two logical values (the orange properties). A first one is used to distinguish solutions of the previous population from those of the current population, while the second one is used to determine if the tDOM-algorithm can be stopped or not. To be able to explain the concept of using the trade-off of a solution as a stopping criterion, the concept of t-domination has to be introduced first.

7.2.1 t-Domination

The concept of non-domination or Pareto-optimality has been mathematically introduced in 2.2. As a reminder: A solution $\mathbf{x}$ is non-dominated or Pareto-optimal if there exists no solution $\mathbf{x}^*$ for which $\forall i \in \{1,\ldots,M\} : f_i(\mathbf{x}^*) \leq f_i(\mathbf{x})$ and $\exists i \in \{1,\ldots,M\} : f_i(\mathbf{x}^*) < f_i(\mathbf{x})$ with $f_i,\ i \in \{1,\ldots,M\}$ the M objective function of the concerned optimisation problem. Figure 7.2(a) graphically represents the concept of non-domination in case of a bi-objective problem. Based on the above non-domination definition, solution p_1 is dominated by solution q_4. q_4 is therefore a non-dominated or Pareto-optimal solution. t-domination however also takes the PIT-region of the solution into account. A solution $\mathbf{z}$ is t-non-dominated if there exists no solution $\mathbf{z}^*$ for which $\forall i \in \{1,\ldots,M\} : f_i(\mathbf{z}^*) \leq f_i(\mathbf{z})$ and $\exists i \in \{1,\ldots,M\} : f_i(\mathbf{z}^*) < f_i(\mathbf{z})$ and $\mathbf{z}^* \notin PIT(\mathbf{z})$, with $f_i,\ i \in \{1,\ldots,M\}$ the M objective function of the concerned optimisation problem. t-domination is graphically represented in Figure 7.2(b) in case of a bi-objective problem. Although q_2 dominates p_1, it does not t-dominates p_1 while it is located in the PIT-region of p_1. q_1 on the other hand does t-dominate p_1, and is therefore t-non-dominated.



(a) Non-domination **(b)** t-domination

Figure 7.2: Graphical representation of non-domination (a) and t-domination (b) in case of a bi-objective problem.

In section 6.2.1 of the previous chapter, it was mentioned that if a solution q is located in the PIT-region of another solution p, there is no significant difference between the two solutions. Because solution q does not display the required trade-off Δt or distribution Δr in comparison to solution p, it does not contribute to the diversification of the solution population, nor does it represent a significantly different process design than solution p. There is therefore no need of keeping q in the solution population and it can be discarded or downgraded.

If the concept of t-domination is extended to the whole solution population, it can be turned into a stopping criterion: if all the solutions of two subsequent solution populations are located in each others PIT-regions, there is no significant difference between the two populations and therefore the algorithm, or iteration process, can be stopped. This is graphically represented in Figure 7.3.

Figure 7.3 displays simultaneously a possible shortcoming of the t-domination stopping criterion. Because all the solutions of the $(t-1)$-th iteration are located in the PIT-regions of the solutions of the t-th generation, the algorithm will be stopped because no new t-non-dominated solutions have been created. The black solutions of the t-th generation represent thus the final solution population. It is clear however that the final solution population has not converged to the Pareto front. Although Figure 7.3 represents a thought experiment, it can be assumed that if the tDOM-algorithm is not capable to produce diverse offspring solutions, the algorithm will be prematurely stopped. If for instance the crossover and mutation parameters are carelessly defined, it can be the case that the offspring solutions are generated within the PIT-region of their parents. And although these solutions might display a slightly better convergence to the Pareto front than their parents, the algorithm does not consider them different enough. Therefore the algorithm will stop, rather than allowing the solutions to creep towards the Pareto front. On the other hand, if the user desires solutions with a high trade-off and distribution, it might also pose difficulties in terms of solution convergence to the Pareto front. The larger the PIT-region of a solution is, the higher the chance that its off-spring solution is located within its PIT-region. In order to rigorously test the performance of the tDOM-algorithm, these assump-

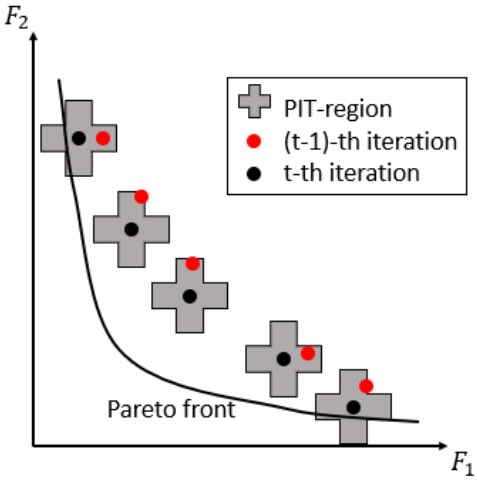


Figure 7.3: t-domination as a stopping criterion.

tions must evaluated. Therefore, in section 7.5, the influence of the crossover, mutation, and trade-off and distribution parameters on the solution convergence and diversity, will be evaluated. Firstly, the adapted trade-off function is presented.

7.2.2 The adapted trade-off function

Algorithm 7 Adapted trade-off function

Require: pop_t, pop_{t-1}, Δt, Δr
  for i=1 **to** $|pop_t|$ **do**
    $pop_t(i)$.TradeOffCounter=0
    $pop_t(i)$.Stop $=$ false
  end for
  $pop_{tot} = pop_t \cup pop_{t-1}$
  $pop_{tot}, \mathcal{F}_{tot} =$ NonDominatedSorting(pop_{tot})
  $pop_{tot,n} =$ NormaliseCosts(pop_{tot})
  for i=1 **to** $|\mathcal{F}_{tot}|$ **do**
    ⋮
    for k=1 **to** |Costs| **do**
      ⋮
      if Costs(k)$\in pop_t$ **then**
        if InPIT(Costs(itDown), Costs(k), Δt, Δr) **then**
          if Costs(itDown)$\in pop_t$ **then**
            $pop(k)$.TradeOffCounter++
          else
            $pop(k)$.Stop $=$ true
          end if
        end if
        ⋮
        if InPIT(Costs(itUp), Costs(k), Δt, Δr) **then**
          if Costs(itUp) $\in pop_t$ **then**
            $pop(k)$.TradeOffCounter++
          else
            $pop(k)$.Stop $=$ true
          end if
        end if
        ⋮
      end if
    end for
  end for

The adapted trade-off function, as presented in Algorithm 7, uses the same framework as the trade-off function as presented in 6.2.2. Only the different properties of the adapted trade-off function will be discussed.

Suppose the algorithm is in its t-th iteration. The adapted trade-off function requires the trade-off parameter Δt, the distribution parameter Δr, and two solution populations: the current solution population pop_t and the previous solution population pop_{t-1}. In the initialisation step, the trade-off counter of the solutions in pop_t is reset to zero. Additionally the stop property of these solutions is set to false. If it is found that a solution of the population pop_{t-1} is located within the PIT-region of a solution of the pop_t population, this stop property will be set to true.

Subsequently the two solution populations are merged into the total solution population pop_{tot}. Because the trade-off counter of solutions is still determined at a speed of one non-dominated front at a time, the population pop_{tot} first has to be sorted in its non-dominating fronts $\mathcal{F}_{tot}$. Also the objective cost of the solutions have to be normalised. Because the best N solutions have yet to be selected from pop_t, it still contains $2N$ solutions, while $|pop_{t-1}| = N$. Because $|pop_{tot}| = 3N$, the non-dominated sorting function and cost normalisation step can be time consuming. From this point, the algorithm continues like the trade-off function as presented in 6.2.2.

If a solution is found to be located in the PIT-region of a concerned solution k, it is verified if the solution is part of pop_t. If this is the case, the trade-off counter of the solution k is incremented with 1. If not however, the stop property of the solution k is set to true. Note that the PIT-region is only constructed for solutions in pop_t.

7.3 The tDOM-evolutionary algorithm

If the adapted trade-off function is completed, all the trade-off counters of the solutions in pop_t will have been determined. The population can than be sorted based on the same principles as presented in 6.4, allowing the selection of the N best solutions. However, in case that all the solutions of pop_t were found to have at least one solutions of pop_{t-1} in their PIT-region, it is no longer necessary to continue the iteration process. The solutions of pop_t are namely not different enough than the solutions of the previous population pop_{t-1}.

Yet, it is again opted to include the FPOS of the solution population as an additional stopping criterion. It was already mentioned that in some cases non-dominated solutions are more easily generated than in others. This is confirmed by the significant difference in needed iterations of the different case studies before the FPOS amounts to 1 (see Figure 5.5 for instance). Including the FPOS as an additional stopping criterion allows the algorithm to at least generate the amount of solutions that is desired by the user. The complete pseudo-codes of the tDOM-algorithms are included in Appendix D.

The tDOM-algorithms are in many aspects the equivalent of the tM-algorithms. The only major difference is that the tDOM-algorithms use a stopping criterion, which remains relevant to the concerned optimisation problem, but does not require extra user-defined parameters. The trade-off parameter Δt and distribution parameter Δr are namely no additional parameters but requirements that are imposed by the user. One of the expected disadvantage of the tDOM-algorithms however is that the value of the trade-off and distribution parameter will have an influence on the convergence of the tDOM-solutions. Additionally it is expected that the EA-parameters which determine the generation of offspring solutions, will also have an influence on the overall performance of the tDOM-algorithms. Lastly it might be possible that the tDOM-algorithm is slower than the tM-algorithm because the time consuming non-dominated sorting function is used on a large solution set.

7.4 Case studies

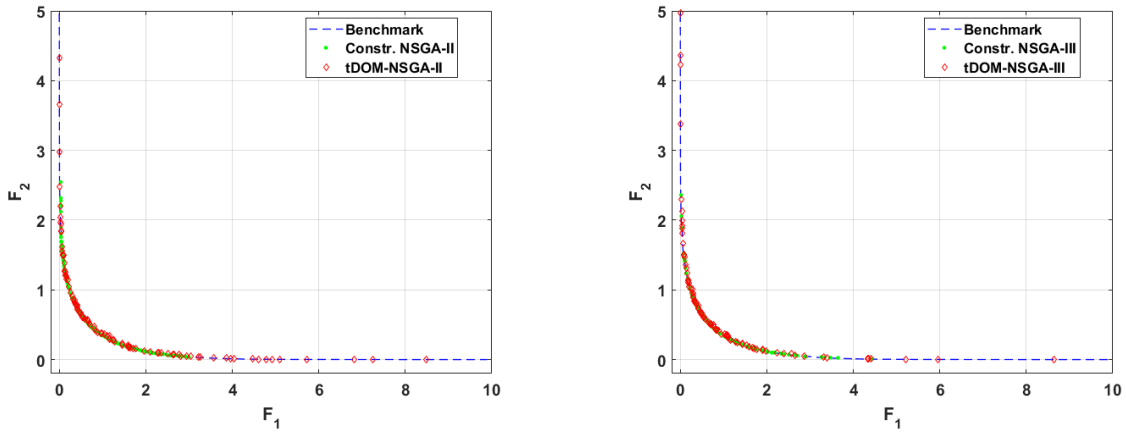
tDOM-NSGA-II and tDOM-NSGA-III are tested on the case studies that were presented in 4.3. The used settings of the EA, trade-off, and distribution parameters, used for each case study, are summarised in Table 6.1. In 7.5, the influence of these settings on the performance of the tDOM-algorithm will be evaluated.

Table 7.1: The EA, trade-off, and distribution parameters setting, used for each case study.

Parameter	Value
Maximum iterations	75
Population size	100
Crossover probability p_c	90.0 %
Mutation probability p_m	10.0 %
Mutation rate μ	5.00 %
Mutation step size σ	$0.05 \times (\mathbf{b} - \mathbf{a})$
Objective subdivisions	99 (*), 13 (**)
Reference points	100 (*), 105 (**)
Trade off Δt	5.00 %
Distribution parameter Δr	10.0 %

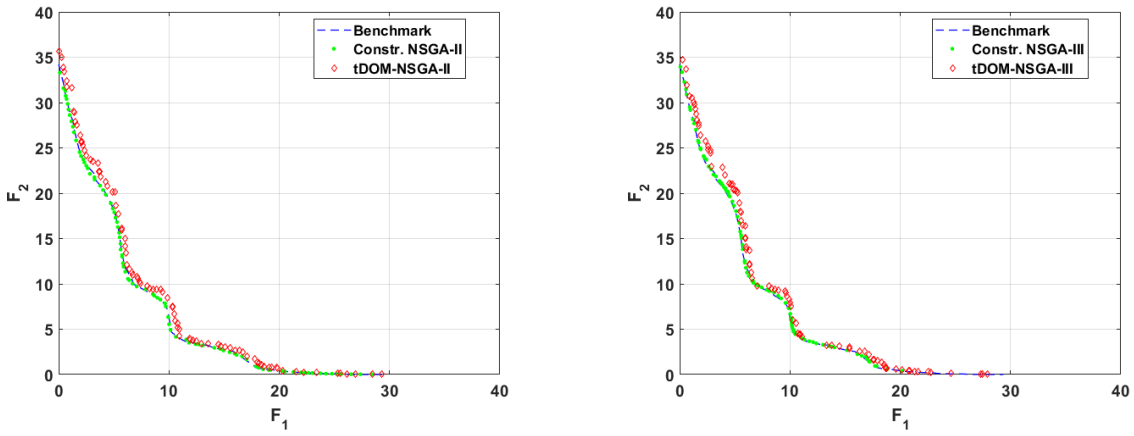
Just like for the constrained-NSGA-II and the tM-NSGA-III algorithm, the number of objective subdivisions d in the tDOM-NSGA-III algorithm is set so that the number of reference points $H \geq N$. The trade-off Δt and distribution parameter Δr have the same value for each objective for the sake of clarity.

7.4.1 The bi-objective case studies



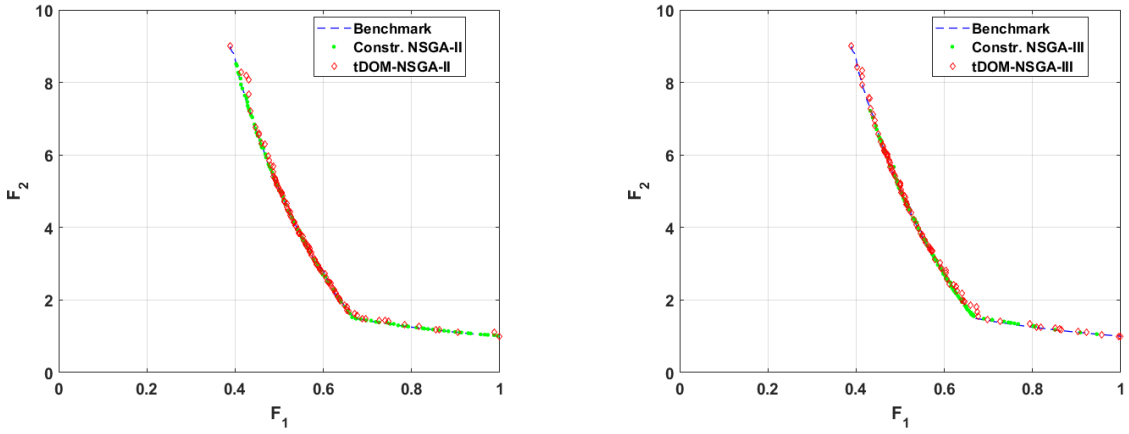
(a) tDOM-NSGA-II **(b)** tDOM-NSGA-III

Figure 7.4: Pareto front of the BIOBJ-problem generated via tDOM-NSGA-II(a) and tDOM-NSGA-III(b).



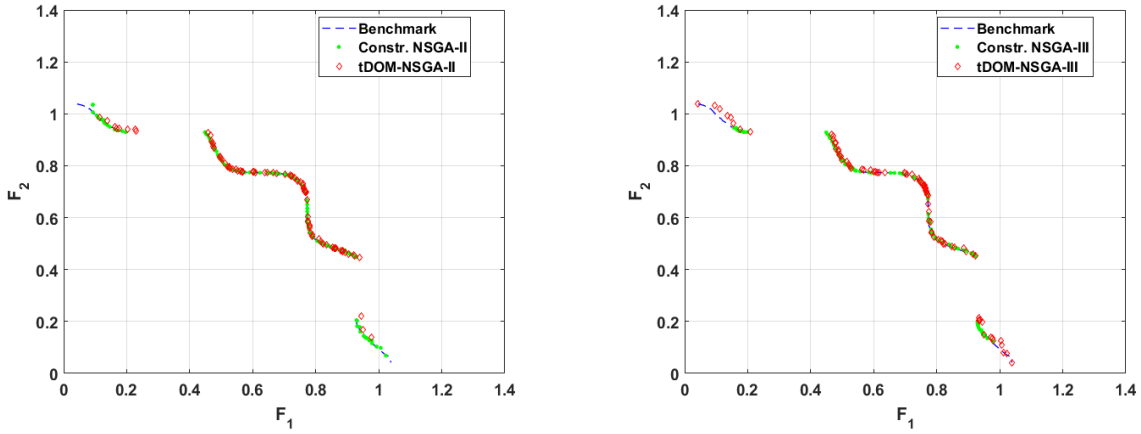
(a) tDOM-NSGA-II **(b)** tDOM-NSGA-III

Figure 7.5: Pareto front of the DO2DK-problem generated via tDOM-NSGA-II(a) and tDOM-NSGA-III(b).



(a) tDOM-NSGA-II **(b)** tDOM-NSGA-III

Figure 7.6: Pareto front of the CONSTR-problem generated via tDOM-NSGA-II(a) and tDOM-NSGA-III(b).



(a) tDOM-NSGA-II **(b)** tDOM-NSGA-III

Figure 7.7: Pareto front of the TNK-problem generated via tDOM-NSGA-II(a) and tDOM-NSGA-III(b).

Figures 7.4 to 7.7 represent the Pareto fronts of the bi-objective case studies that were generated with tDOM-NSGA-II and tDOM-NSGA-III. The generated Pareto fronts allow to draft several conclusion based on a visual comparison with the constrained-solutions and the benchmark Pareto front.

In Figure 7.4, the Pareto fronts of the BIOBJ-problem are represented, generated by tDOM-NSGA-II and tDOM-NSGA-III respectively. The tDOM-solutions of the BIOBJ-problem display a similar convergence to the benchmark Pareto front as the constrained-solutions, but display a higher solution diversity. At first glance, the solution diversity of the tDOM-NSGA-II algorithm seems to be higher than that of the tDOM-NSGA-III algorithm. The performance parameters, which are summarised in Table 7.2, indicate however the opposite scenario but these will be discussed more in detail hereafter. Also, both tDOM-Pareto fronts display the desired trade-off based solution resolution.

The tDOM-Pareto fronts of the DO2DK-problem, represented in Figure 7.5, display a reduction in solution convergence in comparison to the constrained-Pareto front. This reduction was also visible in the tM-Pareto fronts (see Figure 6.5). If the tDOM-Pareto fronts and tM-Pareto fronts of the DO2DK-problem are compared with each other however, it is visible that the tDOM-Pareto front does not as clearly show

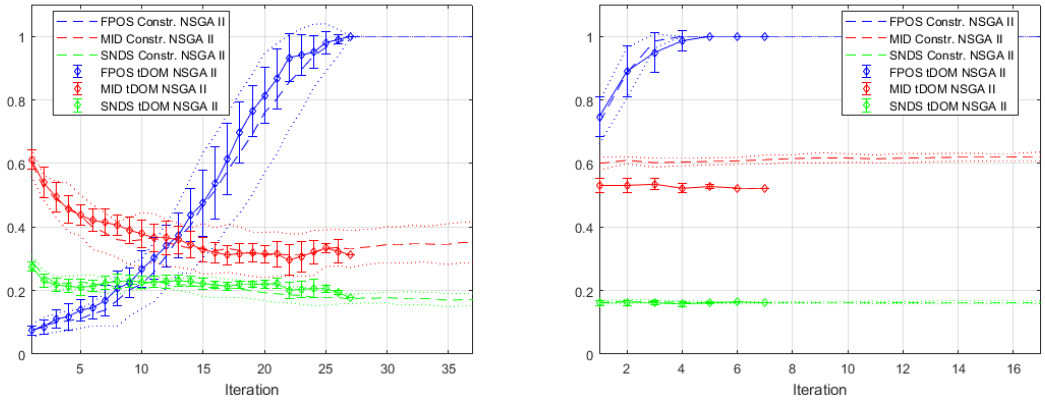
the trade-off based solution resolution as the tM-Pareto front. Because achieving such a trade-off based Pareto front resolution was one of the main targets of the tDOM- and tM-algorithms, this can be seen as a disadvantage of the tDOM-algorithm compared to the tM-algorithm.

The CONSTR-problem does show the desired trade-off based Pareto front resolution but displays a poorer solution convergence in the vicinity of the Pareto front extremities. Although the steep part of the Pareto front, located between the F_1 costs 0.4 and $\pm$0.65, is a high trade-off area, the solution resolution of the tDOM-NSGA-II Pareto front is low between the F_1 costs 0.4 and $\pm$0.475. Apparently the solutions generated by the tDOM-NSGA-II algorithm have converged to the local optima of the Pareto front, which is located in the vicinity of the bend. This problem is not as distinct in case of the tDOM-NSGA-III algorithm. The TNK-problem displays a similar difficulty. The tDOM-Pareto fronts of the TNK-problem namely display an unsatisfactory solution convergence and density in the vicinity of the Pareto front extremities.

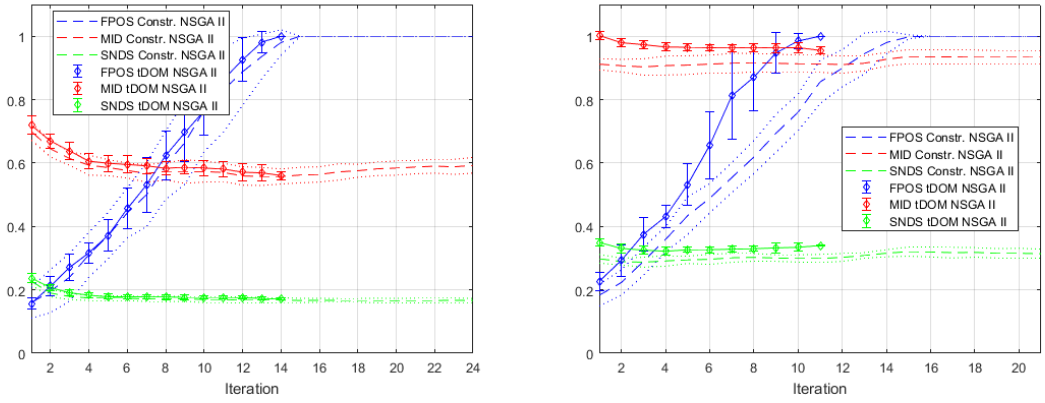
The performance parameters of the tDOM-algorithms are summarised in Table 7.2. The best values of each performance parameter is again compared to the best corresponding value of the constrained-algorithms, and also to that of the tM-algorithms. These ratios are included in the second last and last column respectively. The FPOS performance parameter is not included in Table 7.2, because the value of this performance parameter is fixed by the outlay of the tDOM-algorithms. The performance plots of the tDOM-NSGA-II algorithm are included in Figure 7.8. The performance plots of the tDOM-NSGA-III algorithm for the bi-objective case studies, are included in Appendix D.3.

Table 7.2: Performance parameters and runtime of the tDOM-algorithms for the bi-objective case studies. The best value, per case study, of the concerned performance parameter is set in bold.

	tDOM-NSGA-II		**tDOM-NSGA-III**		**Constr. EA**	**tM-EA**
Performance parameter	**Mean**	**St. dev.**	**Mean**	**St.dev.**	**Relative difference [%]**	**Relative difference [%]**
BIOBJ-problem						
Iterations [-]	23.7	1.7	**22.8**	**2.5**	-69.90	-2.98
MID [-]	**0.2829**	**0.0324**	0.2934	0.0429	-35.37	+22.57
SNDS [-]	0.1828	0.0140	**0.1883**	**0.0276**	+1.18	+6.26
Runtime [s]	**328.4456**	**23.0812**	347.2643	28.9220	+50.72	+341.98
DO2DK-problem						
Iterations [-]	3.8	1.3	**3.6**	**0.8**	-95.20	-18.18
MID [-]	0.5320	0.0192	**0.5271**	**0.0236**	-15.39	+0.67
SNDS [-]	0.1573	0.0069	**0.1611**	**0.0140**	-2.54	+1.32
Runtime [s]	60.3186	17.4929	**58.1949**	**11.7676**	-75.08	+151.93
CONSTR-problem						
Iterations [-]	14.7	5.2	**12.7**	**1.0**	-83.07	-5.22
MID [-]	**0.5693**	**0.0272**	0.5917	0.0165	-17.60	+5.37
SNDS [-]	0.1719	0.0052	**0.1759**	**0.0061**	+1.68	+2.39
Runtime [s]	187.8517	70.1929	**174.0846**	**14.7861**	-13.22	+239.33
TNK-problem						
Iterations [-]	**9.4**	**1.3**	13.2	0.9	-87.47	-32.37
MID [-]	0.9651	0.0131	**0.9442**	**0.0032**	+0.27	+0.00
SNDS [-]	**0.3361**	**0.0100**	0.3138	0.0149	+11.96	+5.13
Runtime [s]	**134.3552**	**23.3241**	194.0388	14.1227	-13.92	+163.54



(a) BIOBJ-problem **(b)** DO2DK-problem



(c) CONSTR-problem **(d)** TNK-problem

Figure 7.8: Comparing performance plots of the tDOM-NSGA-II algorithm and the constrained NSGA-II algorithm (average taken over 10 repetitions). The iteration axis is cropped.

Table 7.2 indicates a decrease in the number of iterations needed in case of the tDOM-algorithm, in comparison both to the constrained-algorithm and the tM-algorithm. However, when the required runtime is compared with the the one of the tM-algorithm, for all the case studies a significant increase is noticed. The tDOM-algorithm even requires a higher runtime than the constrained-algorithm in case of the BIOBJ-problem. This observation confirms the concern that was stated in 7.3. The Matlab software allows the user to time parts of the code, which allows the user to pinpoint time consuming sections in the Matlab code. This is done for the tDOM-NSGA-II algorithm in case of the BIOBJ-problem.

It was found that, on average, the Matlab code of the adapted trade-off function requires 235.9672 s of the total computation time of the algorithm, which is 71.84 % of the total average runtime of the tDOM-algorithm. The non-dominated sorting function is accountable for 95.20 %, on average, of the fraction of the computation time taken up by the adapted trade-off function. These figures clearly indicate that the use of the non-dominated sorting function on the large solution population, is the main decelerator of the tDOM-algorithms. The performance results of the tM-algorithms indicate that the trade-off function in these algorithms (see 6.2.2) is undoubtedly less cumbersome and time consuming than the adapted trade-off function of the tDOM-algorithms. Replacing the adapted trade-off function by the trade-off function as used in the tM-algorithms is therefore highly recommended. This can be achieved by only using the adapted trade-off function if the FPOS additionally is equal to one. Based on the layout of the tDOM-

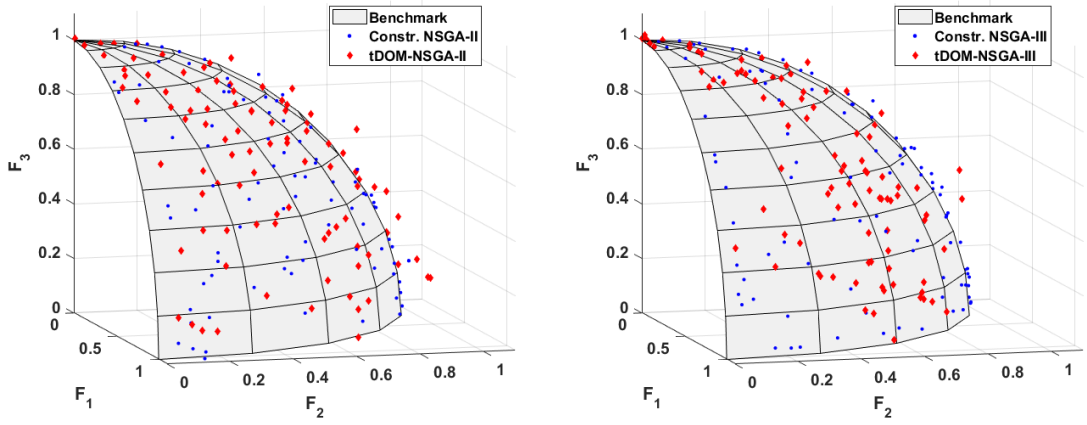
algorithm, then and then alone the algorithm can be stopped (if additionally the solutions that are currently generated do not t-dominate the solutions of the previous iteration). The required runtime of the improved tDOM-algorithms is summarised in Table 7.3 for each bi-objective case study.

Table 7.3: Runtime of the improved tDOM-algorithms and comparisons to the un-improved tDOM-algorithms.

Case study	Improved tDOM-NSGA-II [s]	Decrease [%]	Improved tDOM-NSGA-III [s]	Decrease [%]
BIOBJ	86.5082 $\pm$ 13.3024	-73.66	111.9432 $\pm$ 5.0804	-67.76
DO2DK	22.1332 $\pm$ 8.8652	-63.31	26.7371 $\pm$ 5.3611	-54.06
CONSTR	49.0207 $\pm$ 7.5310	-73.90	63.1692 $\pm$ 10.0027	-63.71
TNK	57.3110 $\pm$ 1.6490	-57.34	71.7257 $\pm$ 8.2896	-63.04

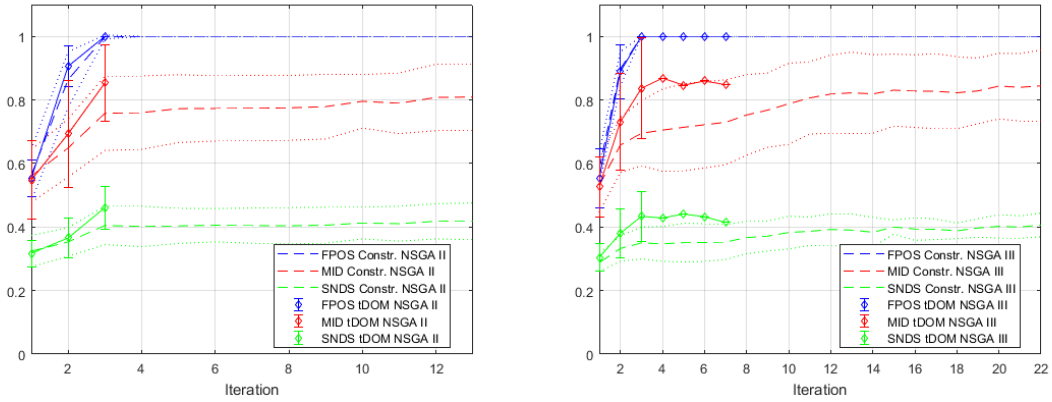
The well-considered use of both versions of the trade-off function indeed results in a significant decrease in required computation time. This is confirmed by the decreases, in terms of percentages, summarised in Table 7.3. For all the tested bi-objective case studies, the decrease in computation time, when using the improved tDOM-algorithm instead of the tDOM-algorithm, amount to more than 50.00 %.

7.4.2 The three-objective case study



(a) tDOM-NSGA-II **(b)** tDOM-NSGA-III

Figure 7.9: Pareto fronts of the DTLZ2.3-problem, generated via tDOM-NSGA-II (a) and tDOM-NSGA-III (b).



(a) tDOM-NSGA-II **(b)** tDOM-NSGA-III

Figure 7.10: Performance plots of the tDOM-NSGA-II and tDOM-NSGA-III algorithms for the DTLZ2.3-problem (average taken over 10 repetitions). The iteration axis is cropped.

Based on the Pareto fronts of the DTLZ2.3-problem (see Figure 7.9), it is obvious that the solution convergence is weak. Based on a visual comparison, this is especially the case for the tDOM-NSGA-II algorithm and low F_3 objective costs. The solution convergence of the tDOM-NSGA-III algorithm is better but the respective performance plot (see Figure 7.10(b)) and MID performance parameter summarised in Table 7.4, contradict this. The increased MID of the tDOM-NSGA-III algorithm can however be partially assigned to an increase in solution diversity. The corresponding performance plot namely indicates that the SNDS performance parameter evolves similarly to the MID performance parameter, with an increasing number of iterations. As previously mentioned, these two performance parameters are correlated to one another.

Table 7.4: Performance parameters and runtime of the tDOM-algorithms for the DTLZ2.3-problem. The best value of the concerned performance parameter is set in bold.

	tDOM-NSGA-II		**tDOM-NSGA-III**		**Constr. EA**	**tM-EA**
Performance parameter	**Mean**	**St. dev.**	**Mean**	**St.dev.**	**Relative difference [%]**	**Relative difference [%]**
Iterations [-]	3.9	3.0	**3.2**	**1.3**	-95.73	-25.58
MID [-]	**0.7214**	**0.0959**	0.8178	0.1407	-20.56	-11.02
SNDS [-]	0.3839	0.0458	**0.4149**	**0.0750**	-9.88	-8.02
Runtime [s]	**45.3282**	**31.9464**	56.7563	22.5728	-79.15	+104.59

On a general note, no other significant differences are noticed between the tM-algorithms and tDOM-algorithms in case of the DTLZ2.3-problem. Thus instead of delving deeper into the results of the tDOM-algorithms for the three-objective case study, it is opted to study the influence of the trade-off parameters Δt and Δr, and the influence of the EA-parameters on the performance of the *improved* tDOM-algorithm.

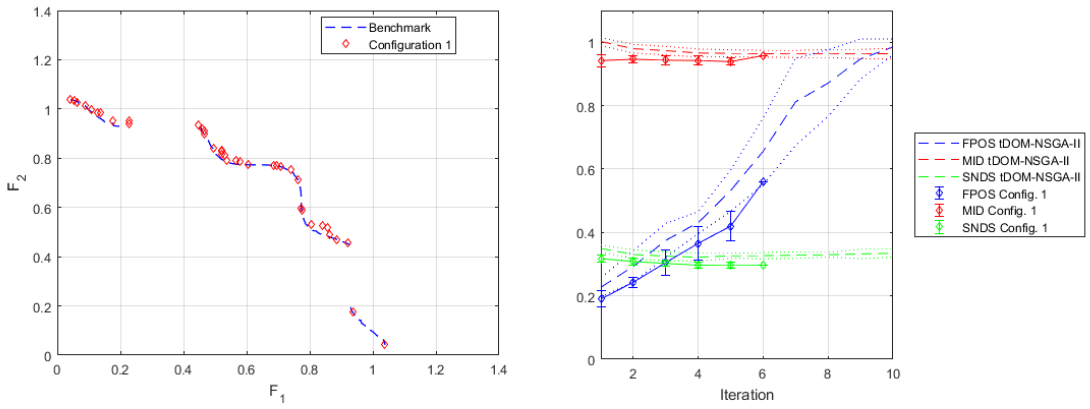
7.5 Influence of the EA- and trade-off parameters

For the sake of clarity, and avoidance of overloading the reader with information, the influence of the EA- and trade-off parameters is only conducted for the *improved* tDOM-NSGA-II algorithm and on one bi-objective case study: the TNK-problem. Based on the results, a recommendation scheme for defining the EA- and trade-off parameters will be drafted.

7.5.1 The EA-parameters

Several parameters are covered by the term 'EA-parameters'. The main interest in this scenario lies with the EA-parameters that control the generation of offspring solutions. The parameters in question are the p_c and p_m parameters on one hand, which define the probability of a parent solution to participate in a crossover or mutation, and the μ and σ parameters on the other hand, which control the mutation rate and magnitude respectively. It is assumed that an ill-considered definition of these parameters can lead to a decreased performance of the improved tDOM-algorithm.

In the tDOM-NSGA-II, a precautionary additional consideration has already been implemented. Using the value of the FPOS performance parameter as an additional stopping criterion prohibits the premature interruption of the algorithm. To thoroughly examine the influence of the EA-parameters, and the trade-off parameters for the same matter, the additional FPOS stopping criterion is removed from the improved tDOM-NSGA-II algorithm's outlay. This additional parameter ensured that the number of generated non-dominated solutions was equal to the desired population size N. It is therefore possible that the results of the benchmark situation will differ from the results presented in 7.4.1 and 7.4.2. The influences of the EA-parameters and trade-off parameters will be compared to this new benchmark. This new benchmark Pareto front is presented in Figure 7.11, accompanied with its performance plot.



(a) Pareto front **(b)** Performance plot

Figure 7.11: Pareto front and performance plot of the TNK-problem, generated with the improved tDOM-NSGA-II algorithm without the FPOS stopping criterion.

The performance plot confirms the expected decrease in FPOS. While the tDOM-NSGA-II algorithm with the additional FPOS stopping criterion was forced to continue the iteration until N non-dominated solutions were generated, the improved tDOM-NSGA-II is not forced to do so and is stopped on a FPOS of 0,4160 $\pm$ 0,0794.

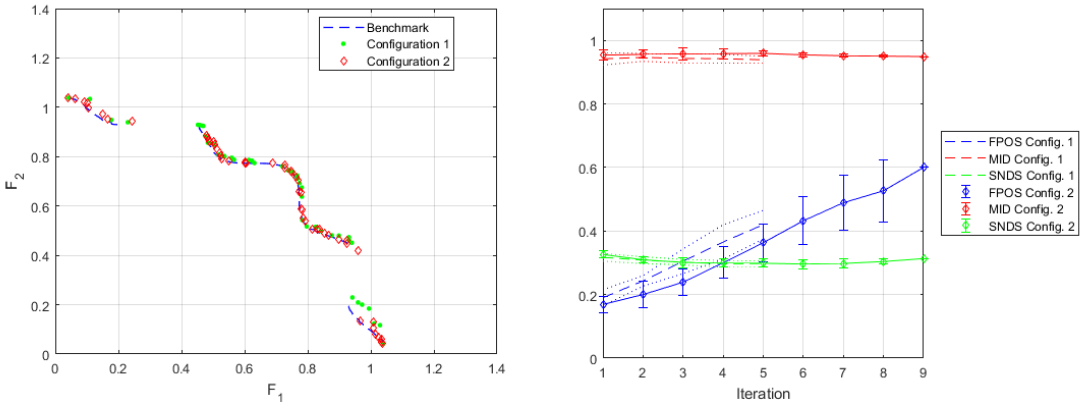
The tested EA-parameter configurations are presented in Table 7.5. The first configuration represents the benchmark situation, which has been used for all the case studies thus far. Configuration 2 represents a situation in which both the crossover and mutation have small values. This configuration is used to simulate a scenario in which the offspring solutions are generated in the vicinity of their parents. While the step size of the crossover is not controllable (see 3.3.2), its probability is halved. The step size of the mutation is controllable and therefore the probability of a mutation is accordingly increased, but its step size is ten times smaller than in the benchmark configuration. Configuration 3 represents a scenario with a large mutation probability which has additionally an increased step size. Lastly, configuration 4 represents

a scenario with an increased crossover probability. The trade-off parameters will be kept constant in all the configurations.

Table 7.5: Test configuration for determining the influence of the EA-parameters on the performance of the improved tDOM-NSGA-II algorithm.

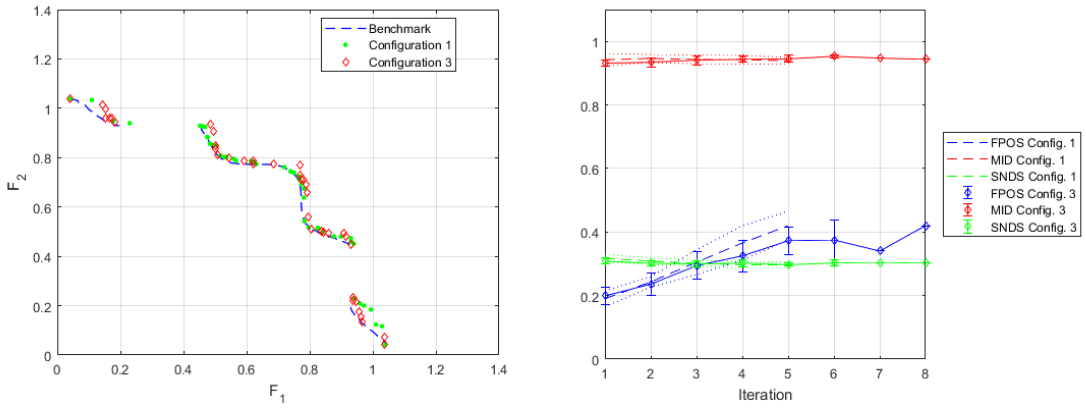
Configuration	p_c	p_m	μ	σ	Δt	Δr
1	0.90	0.10	0.05	$\mu(\mathbf{b}-\mathbf{a})$	0.05	0.10
2	0.45	0.55	0.005	"	"	"
3	0.45	0.55	0.50	"	"	"
4	0.99	0.01	0.05	"	"	"

In Table 7.5, $\mathbf{a}$ and $\mathbf{b}$ respectively represent the lower and upper boundaries of the n decision variables and are thus vectors. The expectation is that if the offspring solutions are generated in the vicinity of the parent solutions, the algorithm will stop more quickly, resulting in a unsatisfactory solution convergence. Configuration 2 represents this scenario. The reasons for this, is explained in 7.2.1. The different Pareto fronts of configurations 2, 3, and 4 are represented in Figure 7.12 up to and including Figure 7.14 with their corresponding performance plot.



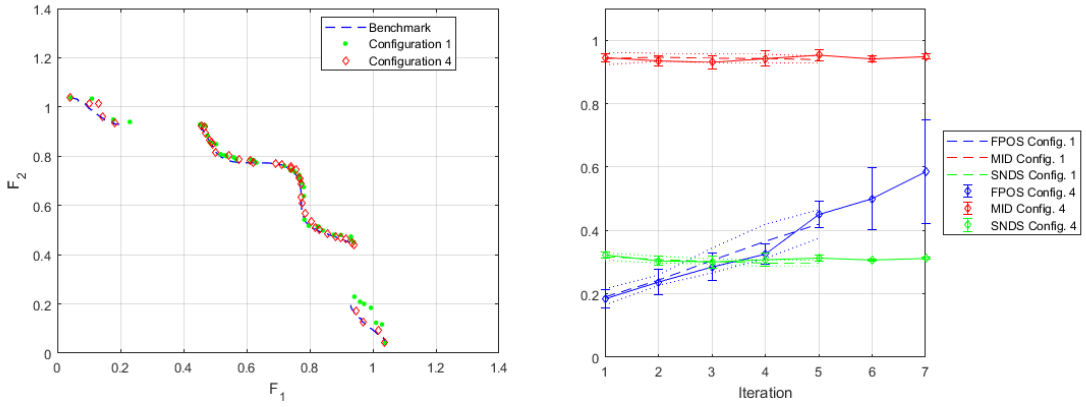
(a) Configuration 2 **(b)** Configuration 2

Figure 7.12: Pareto front and performance plot of configuration 2 of Table 7.5, compared to configuration 1.



(a) Configuration 3 **(b)** Configuration 3

Figure 7.13: Pareto front and performance plot of configuration 3 of Table 7.5, compared to configuration 1.



(a) Configuration 4 **(b)** Configuration 4

Figure 7.14: Pareto front and performance plot of configuration 4 of Table 7.5, compared to configuration 1.

A visual comparison of the Pareto fronts represented in Figure 7.12 up to and including Figure 7.14, does not indicate a noteworthy difference between the tested EA-configuration. The only major difference that is noticeable, is the difference in the number of non-dominated solutions that have been generated. This is also confirmed by the performance plots in Figure 7.12 up to and including Figure 7.14. The third configuration, in which the mutation step dominates the offspring generation, displays the lowest average FPOS value (0.3220 ± 0.0673). Moreover, the average amount of iterations that were needed for this configuration were the lowest of all the configurations. It appears that for this configuration the improved tDOM-NSGA-II algorithm was stopped prematurely. This resulted in a low number of non-dominated solutions that were generated. This result is however contradictory to the expectations. It was expected that configuration 2 would show the worst solution convergence and algorithm performance.

The Pareto fronts of the different EA-configurations, nor the corresponding performance plots indicate a significant difference in performance of the improved tDOM-NSGA-II algorithm. This is an unexpected results while it is sensible to assume that if the offspring solution do not differ much from their parents, they are more likely to be located in the PIT-region of the latter. Yet, bearing in mind that Deb et al. (2002), Valadi and Siarry (2014) and Liagkouras and Metaxiotis (2017) have recommended a crossover probability

of 90.0 %, or a $p_c = 0.90$, a mutation probability p_m of $1/n$, with n the number of process variables, it is most likely that the user will follow these recommendations. It is only more assuring to observe that the settings of the offspring EA-parameter do not influence the performance of the tDOM-NSGA-II algorithm.

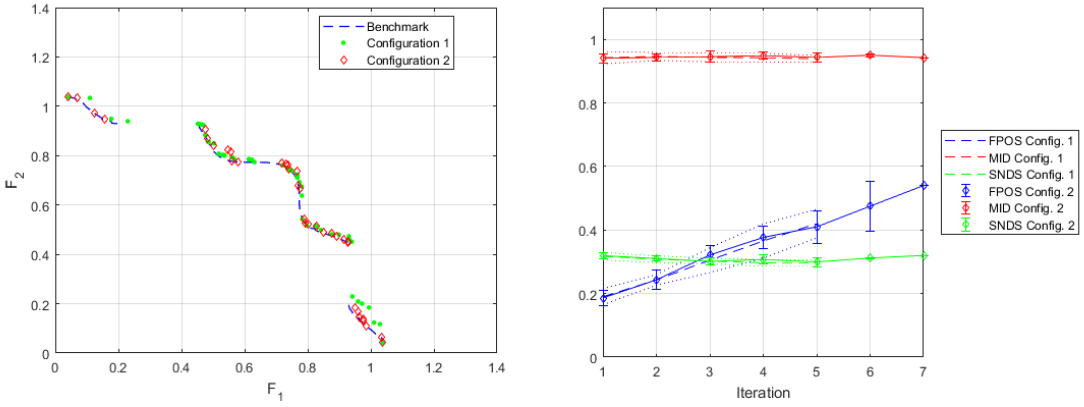
7.5.2 The trade-off parameters

The tested trade-off parameter configurations are presented in Table 7.6. The first configuration is the benchmark configuration, which has been used for all the case studies so far. The second configuration simulates a scenario with an increased trade-off parameter, while the third configuration simulates a scenario with an increased distribution parameter. The last configuration simulates a scenario in which both the trade-off parameter and distribution parameter have increased values.

Table 7.6: Test configurations for determining the influence of the trade-off parameters on the performance of the improved tDOM-NSGA-II algorithm.

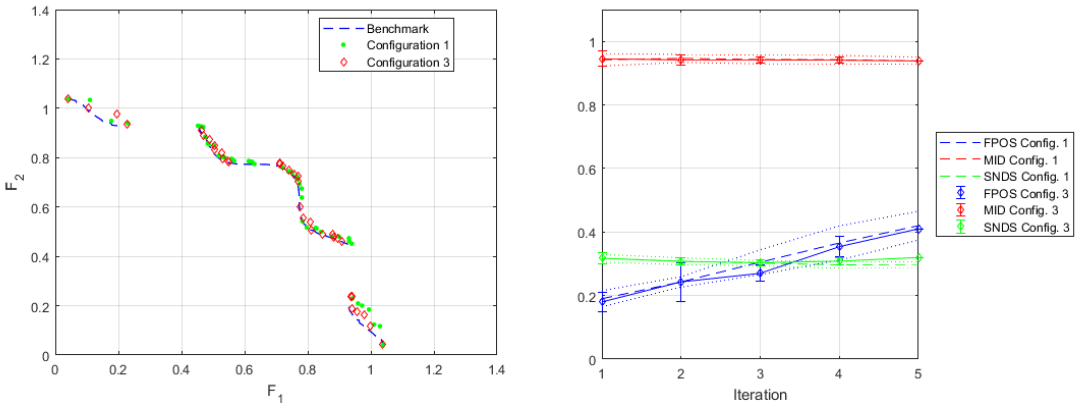
Configuration	p_c	p_m	μ	σ	Δt	Δr
1	0.90	0.10	0.05	$\mu(\mathbf{b}-\mathbf{a})$	0.05	0.10
2	"	"	"	"	0.25	0.10
3	"	"	"	"	0.05	0.50
4	"	"	"	"	0.25	0.50

Again, the expectation is that the improved tDOM-NSGA-II algorithm will display a decrease in overall performance in case of configuration 4. Because in this scenario, the PIT-regions of the solutions are larger, the probability that the parent solutions are located within the PIT-region of their offspring, increases. The generated Pareto fronts, and the corresponding performance plots, of configurations 2 to 4 are presented in Figures 7.15 up to and including Figure 7.17.



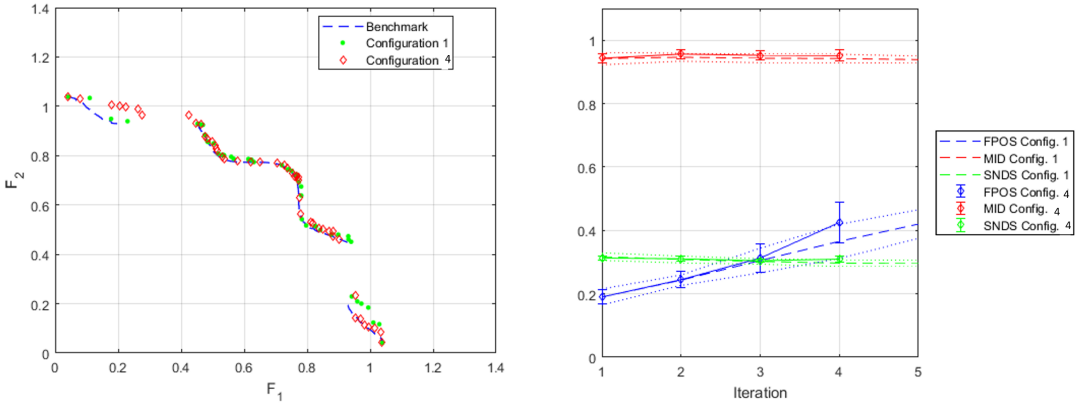
(a) Configuration 2 **(b)** Configuration 2

Figure 7.15: Pareto front and performance plot of configuration 2 of Table 7.6, compared to configuration 1.



(a) Configuration 3 **(b)** Configuration 3

Figure 7.16: Pareto front and performance plot of configuration 3 of Table 7.6, compared to configuration 1.



(a) Configuration 4 **(b)** Configuration 4

Figure 7.17: Pareto front and performance plot of configuration 4 of Table 7.6, compared to configuration 1.

Based on a visual comparison, some differences in the Pareto fronts can be distinguished. When the Pareto fronts of configuration 2 and 3 are compared, it can be seen that the low trade-off areas of the Pareto fronts are distinctively less populated in configuration 3 than in configuration 2. This is because configuration 3 highly stresses on the distribution of the solutions with its increased distribution parameter Δr. As already mentioned in 6.2, the Δr parameter defines how crowded the low trade-off areas of the Pareto front will be. The higher the Δr value, the lower the amount of solutions that will be generated in these low trade-off areas. The Pareto front of configuration 3 affirms this.

A second observation, based on the comparison between configuration 4 on one hand and configuration 2 and 3 on the other hand, is that the improved tDOM-NSGA-II algorithm is less capable to distinguish the trade-off of solutions if the PIT-region becomes larger. This statement is made based on the loss of the trade-off based Pareto front resolution in configuration 4. If PIT-regions become larger, chances are that *all* the generated solutions have a high trade-off counter and the differences in high and low trade-off solutions becomes less distinct. Additionally the Pareto front of configuration 4 indeed displays the worst solution convergence of all the configurations, so the expectation that large PIT-regions will have a negative effect on the convergence of the final solution population to the Pareto front, is legit. However, if the

user desires such high trade-off and distributions in solutions, he/she also allows more tolerance to the convergence of the solutions. This is graphically represented in Figure 7.18.

If the PIT-region is constructed around solutions that have converged to the Pareto front, a so-called zone of insignificance is formed. The four solutions displayed have indeed converged to the Pareto front, but the solutions that are located in their PIT-region are not considered to be different by the user. If the number of solutions is low, than the PIT-region form distinct regions of insignificance. However, if the number of solutions increases, the PIT-regions of the solutions that have converged to the Pareto front will overlap and will eventually form a band or zone of insignificance. Any solutions within this zone is insignificantly different to the solutions that have actually converged to the Pareto front. This zone becomes naturally increasingly large if the user defines Δt and Δr parameters with high values. This means that solutions which are increasingly further away from the Pareto front, are as valuable to the user as solutions that have fully converged. This leads to the conclusion that the lack of solution convergence in configuration 4 does not pose as a problem, as long as the generated non-dominated solutions have converged within the zone of insignificance. Note that although the visual representation of the Pareto fronts gives rise to the conclusion made above, no real difference in average performance is witnessed based on the performance plots.

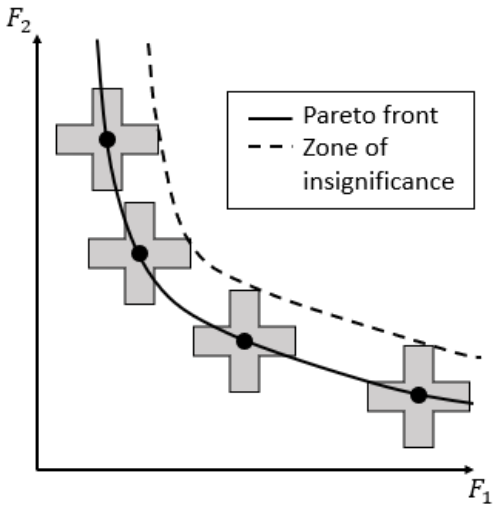


Figure 7.18: Graphical representation of the tolerance on the convergence of solutions with a high Δt and high Δr in case of a bi-objective optimisation problem.

7.5.3 Recommendations

Based on the made EA- and trade-off parameter sensitivity analysis, one major recommendation can be drafted when considering the definition of the tested parameters, which is the following: if the user is aware of the meaning and use of the defined parameters, and the assigned values are based on founded arguments, no decrease in performance is witnessed.

Concerning the EA-parameters, it was nonetheless unexpectedly found that the values of the p_c-, p_m-, μ- and subsequently σ-parameters do not have a major influence on the algorithms performance or the convergence of the solutions to the Pareto front. If the recommendations of Deb et al. (2002), Valadi and Siarry (2014), and Liagkouras and Metaxiotis (2017) are followed for the p_c- and p_m-parameters, the algorithm performs satisfactorily, even with the abolishment of the additional FPOS stopping criterion. The only performance change that was noticed, was a small difference in the number of non-dominated solutions that were generated in the four configurations. However, on a more general note, it can be stated that the improved tDOM-NSGA-II algorithm displays little dependency on the value of the EA-parameters concerning the formation of offspring solutions.

On the contrary, the performance of the improved tDOM-NSGA-II algorithm was found to be more dependent on the values of the trade-off parameters Δt and Δr. It was found that the Δr value had an influence on the crowdedness of the low trade-off areas of the Pareto front. But because this is additionally the

major purpose of this parameter, this came as no surprise. When both parameters became increasingly large, it was apparent that the algorithm had more difficulty in distinguishing low trade-off solutions from high trade-off solutions, which eventually resulted in a loss of the trade-off based Pareto front resolution. The convergence of the solutions to the Pareto front was simultaneously found to be worst in this scenario (i.e. configuration 4). However, because the user desires such high trade-off's Δt and distributions Δr, he/she also allows more tolerance on the convergence of the solutions to the Pareto front. This was demonstrated via the zone of insignificance in Figure 7.18. The user should therefore be aware that with increasing trade-off parameters, the trade-off based resolution of the Pareto front will diminish and the convergence of the solutions to Pareto front will decrease. If the high trade-off parameters are imposed based on founded reasons, these two phenomena are however not a decrease in performance of the algorithm, but rather the inherent results of the imposed trade-off parameters.

7.6 Conclusion

The tM-algorithm, which was introduced in the previous chapter, was a big step forward when compared to the constrained-algorithms of Chapter 5. The introduction of a trade-off function allowed for the generation of Pareto front with a trade-off based solution resolution without the generation of futile solutions. An additional major advantage was the introduction of a problem-related stopping criterion: the MID-stopping criterion. This stopping criterion was based on the elitism of the NSGA-II and NSGA-III algorithms. Because the best solutions of the previous iteration are kept unchanged in the current one, it can be assumed that once the solutions have converged to the Pareto front, they will not move away from it. If however in that scenario the iteration process is not interrupted, new solutions will be generated on the Pareto front itself. The overall convergence however will not change, resulting in a stabilisation of the MID performance parameter. The tM-algorithm was aborted if the difference in MID between two subsequent solution population was smaller than a pre-defined ΔMID tolerance. This ΔMID value had to defined by the user.

The major disadvantage of this stopping criterion is that the user often does not have a notion of what a good solution convergence is for his/her concerned optimisation problem. The user is thus unable to define what an acceptable convergence tolerance is. The ΔMID value is therefore again an arbitrarily defined parameter, just as this was the case when the maximum number of iterations was used as a stopping criterion. The only parameters of which the user often does have a founded notion, are the trade-off and distribution parameters.

It was concluded that the ΔMID value was an unnecessary additional performance parameter. Thanks to the implemented trade-off function, the algorithm is now able to determine the value of the generated solutions for the user. If solutions are densely located within each other PIT-regions, the user will not see them as different from each other. By extrapolating this to two subsequent solution populations, it can be said that if the solutions of two subsequent populations are located within each others PIT-regions, they are not different from each other. This scenario will only happen if the solutions have sufficiently converged to the Pareto front therefore allowing the termination of the iteration process. This idea was also presented by Hashem et al. (2017) in their ”divide and conquer” scheme.

The stopping criterion of the tDOM-algorithms can be formulated as follows: If no solution of the current population t-dominates a solution of the previous population, then the two population are not distinguishable from each other and the algorithm can therefore be stopped. The evaluation of the t-domination of the solutions is done via the adapted trade-off function. This trade-off function has a bivalent use while it calculates the trade-off counter of the solutions of the current iteration. On the other hand, it also sets a solution inherent logic value to true if a solution of the previous iteration is located in the PIT-region of a

solution of the current iteration. If for all the solutions of the current iteration this solutions inherent logic value has been set to true, the algorithm is stopped.

Using t-domination as a stopping criterion in fact abolishes the need for defining a maximum number of iterations. It is opted not to do this, to avoid the odd chance that the algorithm indefinitely continues iterating. Additionally the FPOS performance parameter is again incorporated as a second stopping criterion in order to assure the generation of the desired N non-dominated solutions.

The case studies display however one major flaw of the t-domination stopping criterion and this is that the adapted trade-off function is cumbersome. It requires the non-dominated sorting of solution set with a size of $3N$, which takes up most the computation time for which the adapted trade-off function is accountable for. Because of the additional FPOS stopping criterion, the algorithm can only be stopped if the FPOS of the solution population equals 1. It is only in this case rational to use the cumbersome adapted trade-off function in order to generate the second, trade-off based, stopping criterion. In all other cases, the faster trade-off function, as used in the tM-algorithms, is used. This resulted in a significant time gain in comparison to the original tDOM-algorithms.

The sensitivity analysis of the improved tDOM-NSGA-II algorithm led to one striking result, which was that the overall performance of the algorithm was not dependent on the value of the EA-parameters concerning the generation of offspring solutions. It was expected that if these parameters are ill-chosen and the offspring solutions are generated in the vicinity of their parents, the algorithm would prematurely stop, resulting in a bad solution convergence. Based on three test configurations, which all represented a different scenario concerning the generation of offspring solutions, it was found that this was not the case. Bearing in mind that the user is most likely to follow the recommended EA-parameter settings proposed by Deb et al. (2002), Valadi and Siarry (2014), and Liagkouras and Metaxiotis (2017), it is concluded that the EA-parameters have no major influence on the performance of the improved tDOM-NSGA-II algorithm.

The sensitivity analysis of the improved tDOM-NSGA-II algorithm for the trade-off parameters however showed a different scenario. With increasing Δr values, the low trade-off areas of the Pareto front become increasingly less populated. Because the Δr parameter is used to control this, the stated result comes as no surprise. Nonetheless, the user should be aware of this consequence. Additionally, the algorithm had more difficulties distinguishing between high trade-off and low trade-off solutions if both Δt and Δr, or the PIT-region, became increasingly large. In this scenario, the solution convergence also decreased as a result of the increased tolerance the user consequently allows by defining large trade-off parameters. Again, this not considered to be a decrease in performance, but rather an inherent result of the selected trade-off parameters. The main conclusion that resulted from the sensitivity analysis is that user should always be aware what the consequence are of the parameter values that he/she selects.

Part V

Conclusion

Chapter 8

Conclusions and perspectives for further research

8.1 Conclusion

The presented thesis has been made in view of the increasing need for an optimised process industry. The climate treaty of Paris does not only pose big challenges to governments worldwide, but also to the industry. Process and R&D-engineers in any industry, and in any place, have never been thus challenged to develop new and sustainable manufacturing processes and to improve existing ones. These optimisations often involve multiple objectives that have to be simultaneously met. Mostly these objectives relate to the energy consumption of the process, the yield, and the overall cost. However, these objectives often have contradictory optima and thus the minimisation of a multi-objective optimisation problem (MOOP) has an infinite number of solutions. These solutions are located on the so-called Pareto front. But because solving a MOOP involves complex mathematics, the urge for a tool which allows users to optimise a process quickly and efficiently, is ever increasing.

In the past, several successful attempts have already been developed. Deterministic algorithms, like Weighted sum (see 2.3.1), Normal boundary intersect (see 2.3.2), and Normalised normal constraint (see 2.3.3), are popular solving methods. Deterministic algorithms convert the multi-objective optimisation problem into a set of parameterised single-objective optimisation problems (SOOP) which are subsequently individually solved. This renders one solution per SOOP, or iteration of the deterministic algorithm. This however is one of the disadvantages of the deterministic algorithms, together with the fact that the generated solutions are often prone to converge to local optima, do not display a uniform spread, and non-convex areas of the Pareto front can not be reached in certain cases.

The main focus of this thesis has been on the evolutionary algorithms Non-dominated sorting genetic algorithm-II (NSGA-II) and Non-dominated sorting genetic algorithm-III (NSGA-III). Evolutionary algorithms are a subcategory of the stochastic algorithms, which are in fact the counterparts of the deterministic algorithms. Stochastic algorithms tackle the MOOP in its entirety, are capable of generating multiple solutions per iterations run, and do not tend to converge to local optima. The basic philosophy of the algorithms is based on biological reproduction and evolution. Just like in nature, new solutions are created from parent solutions and by only selecting the best solutions for further reproduction, the generated solutions eventually converge to the most optimal solutions of the MOOP, or the Pareto front.

NSGA-II and NSGA-III are developed by Deb et al. (2002) and Deb and Jain (2014) respectively and are

wildly acclaimed by researchers and users alike. Based on the results of six numerical case studies, it was found however that the algorithms lack the ability to distinguish high trade-off solutions from low trade-off solutions. The former are of higher value to the user, so it is desirable to present the user with a Pareto front that only contains solutions that display a minimal trade-off and distribution (imposed by the user). These high trade-off solutions are located in the steep sections of the Pareto front. Additionally it was found that the default stopping criterion proposed by Deb et al. (2002) and Deb and Jain (2014) lacked problem relevance. The NSGA-II and NSGA-III algorithms were stopped when a pre-defined number of iterations was reached. This parameter was defined by the user, but the irrelevance of it according to the concerned optimisation problem, makes it impossible for the user to meticulously define a suitable value. Often, the stopping criterion will be overestimated by the user, resulting in a significant loss of computation time.

In a first attempt to resolve these two major shortcomings, the tM-algorithms were developed. These algorithms are capable to distinguish low trade-off solutions from high trade-off solutions via the construction of regions of practical insignificant trade-off (PIT-region) around the generated solutions. The PIT-region is introduced by Mattson et al. (2014) and is defined by two trade-off parameters: the trade-off Δt and distribution Δr. Solutions that are located within the PIT-region of a different solution do not display the required trade-off or distribution, imposed by the user, and are therefore considered as indistinguishable from one another. These solutions do not contribute to the diversity of the population, nor do they represent a distinct process design or MOOP-solution, so they can be discarded from the solution population. In the tM-algorithms, the trade-off based Pareto front solution is however achieved without the excessive generation of solutions and this is a novelty for evolutionary algorithms.

The default stopping criterion of reaching a pre-defined number of iterations, was in the tM-algorithms replaced by the MID-criterion. The mean ideal distance, or MID, is a performance parameters used to quantify the convergence of the generated solutions to the Pareto front. It is the mean Euclidean distance between the generated solutions and the Utopia point in the normalised objective space. While NSGA-II and NSGA-III are both elitist algorithms, solutions will not move away from the Pareto front once they have converged to it. Therefore, if the MID of two subsequently generated solution populations becomes smaller than a user-defined tolerance of ΔMID, than the algorithm can be stopped.

The tM-algorithms displayed a significant gain in computation time and the Pareto fronts indeed displayed the desired trade-off based solution resolution. However, the ΔMID stopping criterion was still deemed to be arbitrary. Although it was more relevant to the concerned optimisation problem, the definition of the value of the ΔMID stopping criterion was still mainly based on guessing. The user namely has often no clue what a good solution convergence is for his/her concerned optimisation problem, and can therefore impossibly define an acceptable tolerance in convergence (i.e. the ΔMID value). The user mainly only has founded arguments for the parameter values of the trade-off Δt and the distribution Δr. This conclusion led to the eventual development of the tDOM-algorithms.

The name ”tDOM” is derived from the term ”t-domination”, which is a variant of the non-domination of solutions, as used in NSGA-II and NSGA-III, that also takes the PIT-region of a solution into account. A solution q t-dominates a solutions p, if it dominates solutions p and is located outside the PIT-region of solution p. Using t-domination is legitimised by the common knowledge that if two subsequently generated solution population are not significantly different, the algorithm can be stopped. This can be done because the elitism of the algorithms allows to assume that in that scenario, the Pareto front has been reached. The PIT-region of solutions defines when the user deems two solutions significantly different. So if the solutions of two subsequently generated population are located within each other PIT-region, the user will

not see them as different, so the algorithm can be stopped. The key properties of the constrained-, tM-, and tDOM-algorithms are summarised in Table 8.1.

Table 8.1: Summary of the key properties of the constrained-, tM-, and tDOM-algorithms.

Constrained-algorithms	tM-algorithms	tDOM-algorithms
No trade-off based Pareto front resolution	Trade-off based Pareto front resolution	Trade-off based Pareto front resolution
Problem-irrelevant stopping criterion	Problem-relevant stopping criterion	Problem-relevant stopping criterion
Arbitrary stopping criterion	Arbitrary stopping criterion	Non-arbitrary stopping criterion
High computation time	Low computation time	Low computation time

The tDOM-algorithms show very promising results, but it was feared that the definition of the EA-parameters controlling the generation of offspring, and the trade-off parameters would have a major influence on the algorithms performance. It was assumed that if solutions are generated within the vicinity of their parents, or if the PIT-regions would become increasingly large, the tDOM-algorithm would under-perform. A sensitivity analysis demonstrated however that this was not the case. As long as the user is aware of the inherent consequence the definition of an EA- or trade-off parameter can have on the performance of the algorithm and the display of the Pareto front, no decrease in performance is witnessed.

The novel tDOM-algorithms display a high potential to solve practical optimisation problems in which the trade-off and distribution of solutions are also a key additional requirement. Their parameterless and problem relevant stopping criterion makes them even more attractive. Because they are developed on a fundamental basis, not focussing on one specific branch within engineering, they can be used to solve any engineering problem.

8.2 Further research

Viewed from a chemical and biochemical point of view, it would be convenient to link the Matlab codes of the tDOM-algorithms to Aspen Plus. Aspen Plus is a widely used chemical process simulator, which also allows for the optimisation of processes. The Matlab codes of the tDOM-algorithms can be linked to Aspen Plus via INPROP, which is an interface between Matlab and Aspen Plus, developed by Munoz Lopez et al. (2018). By doing so, the tDOM-algorithms could be used to optimise chemical and biochemical process that are simulated in Aspen Plus.

Although the tDOM-algorithms already show great potential, they can further improved. For instance, it was already touched upon in 6.4 that a structured initial solution population could be beneficial for increasing the speed at which non-dominated solutions are generated, but also for the diversity of the final solution population. It is namely assumed that a diverse initial solution population will more quickly give rise to the generation of a diverse final solution population.

On a more fundamental note, when the many-objective numerical case study was discussed, it was concluded that many-objective case study have little practical relevance. The decision maker, or user of the algorithm, most likely desires a graphical representation of the Pareto front. While this cannot be intuitively done for optimisation problems with four or more objectives, it was stated that in these scenarios it was more sensible to reduce the number of objectives to $\leq$3. This could be done via a sensitivity analysis in

which it is conducted how big the individual influence of the concerned objectives on the overall process is. Objectives that have little influence on the eventual outcome of the overall process, can be ignored when solving the MOOP. As already mentioned in 5.3.3, objective reduction poses a complete new set of challenges that still have to be dealt with.

In the broader spectrum of multi-objective optimisation, it could be beneficial to emphasise more on the interactive visualisation of Pareto fronts. The optimisation process namely does not stop by generating a Pareto front with a certain solution resolution. It is often so that the user is not interested in the complete representation of the Pareto front, but rather in one particular solution or area. An interactive Pareto front could aid the user in his/her decision making process. In that scenario, the user can guide the algorithm to certain areas of the Pareto front of his/her preference.

Part VI

References

Bibliography

H. Asefi, F. Jolai, M. Rabiee, and M.E. Tayebi Araghi. A hybrid nsga-ii and vns for solving a bi-objective no-wait flexible flowshop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 75:1017–1033, 2014.

T. Back. *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, 198 Madison Avenue, New York, New York 10016, 1996.

J. Branke, K. Deb, H. Dierolf, and M. Osswald. Finding knees in multi-objective optimization. In *Parallel Problem Solving from Nature* - PPSN-VIII, pages 722–731, 2004.

CEFIC. Landscape of the european chemical industry 2017, belgium. Online, 2017. URL `https://www.chemlandscape.cefic.org/country/belgium/`.

I. Das and J.E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multi-criteria optimization problems. *Structural Optimization*, 14, 1997.

I. Das and J.E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8, 1998.

K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *Evolutionary Computation,* IEEE *Transactions on*, 18(4):577–601, 2014.

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

K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. *Scalable Test Problems for Evolutionary Multiobjective Optimization*, pages 105–145. Springer London, London, 2005.

I. Hashem, D. Telen, P. Nimmegeers, F. Logist, and J. Van Impe. A novel algorithm for fast representation of a pareto front with adaptive resolution: Application to multi-objective optimisation of a chemical reactor. *Computers and Chemical Engineering*, 106, 2017.

H. Jain and K. Deb. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. *Evolutionary Computation,* IEEE *Transactions on*, 18(4):602–622, 2014.

S.M. Kalami. NSGA-II in MATLAB. Online, 2015. URL `http://yarpiz.com/56/ypea120-nsga2`.

S.M. Kalami. NSGA-III: Non-dominated sorting genetic algorithm, the third version. Online, 2016. URL `http://yarpiz.com/456/ypea126-nsga3`.

K. Liagkouras and K. Metaxiotis. Enhancing the performance of moeas: an experimental presentation of a new fitness guided mutation operator. *Journal of Experimental and Theoretical Artificial Intelligence*, 29(1):91–131, 2017.

F. Logist, B. Houska, M. Diehl, and J. Van Impe. Fast pareto set generation for nonlinear optimal control problems with multiple objectives. *Structural and Multidisciplinary Optimization*, 42:591–603, 2010.

C.A. Mattson, A.A. Mullur, and A. Messac. Smart pareto filter: obtaining a minimal representation of multiobjective design space. *Engineering Optimization*, 36(6):721–740, 2014.

A. Messac, A. Ismail-Yahaya, and C.A. Mattson. The normalized normal constraint method for generating the pareto frontier. *Structural and Multidisciplinary Optimization*, 25, 2003.

C.A. Munoz Lopez, D. Telen, P. Nimmegeers, L. Cabianca, F. Logist, and J. Van Impe. A process simulator interface for multiobjective optimization of chemical processes. *Computers and Chemical Engineering*, 109:119–137, 2018.

M. Rabiee, M. Zandieh, and P. Ramezani. Bi-objective partial flexible job shop scheduling problem: Nsga-ii, nrga, moga and paes approaches. *International Journal of Production Research*, 50(24):7327–7342, 2012.

M. Tanaka, H. Watanabe, Y. Furukawa, and T. Tanino. Ga-based decision support system for multi-criteria optimization. *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, 5:1556–1561, 1995.

J. Valadi and P. Siarry, editors. *Applications of Metaheuristics in Process Engineering*. Springer International Publishing Switzerland, 2014.

M. Vallerio, D. Vercammen, J. Van Impe, and F. Logist. Interactive NBI and (E)NNC methods for the progressive exploration of the criteria space in multi-objective optimization and optimal control. *Computers and Chemical Engineering*, 82, 2015.

Y. Yuan, H. Xu, and B. Wang. An improved NSGA-III procedure for evolutionary many-objective optimization. In GECCO*'14 Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 661–668, 2014.

Part VII

Appendix

Appendix A

MATLAB code

A.1 MID Matlab code

```
function [MID,Mini,Maxi]=CalculateMID(pop)
    % Preamble
    n=numel(pop);
    Cost=[pop.Cost];
    M=size(Cost,1);
    Mini=zeros(M,1);
    Maxi=zeros(M,1);
    % Normalisation
    for i=1:M
        SortMatrix=Cost(i,:);
        [SortedMatrix,¬]=sort(SortMatrix);
        Mini(i)=SortedMatrix(1);
        Maxi(i)=SortedMatrix(end);
    end
    for i=1:n
        for j=1:M
            Cost(j,i)=(Cost(j,i)-Mini(j))/(Maxi(j)-Mini(j));
        end
    end
    % MID calculation
    MID = 0;
    for i=1:n
        c = 0;
        for j=1:M
            c=c+(Cost(j,i)^2);
        end
        MID = MID + sqrt(c);
    end
    MID = MID/n;
end
```

A.2 SNDS Matlab code

```
function SNDS = CalculateSNDS(F1,MID,Min,Max)
    % Preamble
    n=numel(F1);
    Cost=[F1.Cost];
```

```
    M=size(Cost,1);
    d=0;
    % Normalisation
    for i=1:n
        for j=1:M
            Cost(j,i)=(Cost(j,i)-Min(j))/(Max(j)-Min(j));
        end
    end
    % SNDS calculation
    for i=1:n
        c=0;
        for j=1:M
            c=c+Cost(j,i);
        end
        d=d+((MID-c)^2);
    end
    SNDS=sqrt(d/n);
end
```

A.3 Feasibility checks

A.3.1 Feasibility check 1

```
function x=FeasibilityCheck1(x,LowerBound,UpperBound,VarSize,nonlincnstr)

[c,ceq]=nonlincnstr(x);
Lower=any((x≥LowerBound)==0);
Upper=any((x≤UpperBound)==0);
while Lower || Upper || any(ceq) || any(c>0)
    x=unifrnd(LowerBound,UpperBound,VarSize);
    Lower=any((x≥LowerBound)==0);
    Upper=any((x≤UpperBound)==0);
    [c,ceq]=nonlincnstr(x);
end
```

A.3.2 Feasibility check 2

```
function yesno = FeasibilityCheck2(x,LowerBound,UpperBound,nonlincnstr)

[c,ceq]=nonlincnstr(x);
Lower=any((x≥LowerBound)==0);
Upper=any((x≤UpperBound)==0);
if Lower || Upper || any(ceq) || any(c>0)
    yesno=true;
else
    yesno=false;
end
```

A.4 Trade-off function

```
function pop=CalcTradeOff(pop,F,trade,spacer)

    % Initialisation
    M=numel(pop(1).Cost);
    n=numel(pop);
```

```
    for i=1:n
        pop(i).TradeOff=0;
    end

    % Normalisation of objectives
    Cost=[pop.Cost];
    CostSorted=zeros(M,n);
    IndexSorted=zeros(M,n);
    Index=1:n;
    Minima=zeros(M,1);
    Maxima=zeros(M,1);
    Range=zeros(M,1);
    for i=1:M
        [CostSorted(i,:),IndexSorted(i,:)]=sort(Cost(i,:));
        Minima(i)=CostSorted(i,1);
        Maxima(i)=CostSorted(i,end);
        Range(i)=abs(Maxima(i)-Minima(i));
    end
    for i=1:M
        for j=1:n
            Cost(i,j)=(Cost(i,j)-Minima(i))/Range(i);
        end
    end

    % Trade-off calculation
    k=numel(F);
    for i=1:k % for all non-dominated fronts
        Front_Cost=Cost(:,F{k}); % select needed costs
        Front_Index=Index(F{k}); % repmat with indexes in pop of selected solutions
        for j=1:M
            [¬,Sorter]=sort(Front_Cost(j,:));
            Index_sorted_acc=Front_Index(Sorter);
            Front_cost_sorted_obj=Front_Cost(:,Sorter);
            % for a solution in that front
            k_front=numel(F{k});
            for l=1:k_front
                iterleft=1;
                iterright=1;
                while l-iterright>0
                    A=abs(Front_cost_sorted_obj(:,l)-...
                        Front_cost_sorted_obj(:,l-iterright));
                    if any(A<trade) || A(j)<spacer
                        pop(Index_sorted_acc(l)).TradeOff...
                            =pop(Index_sorted_acc(l)).TradeOff+1;
                    end
                    iterright=iterright+1;
                end
                while l+iterleft<k_front
                    A=abs(Front_cost_sorted_obj(:,l)-...
                        Front_cost_sorted_obj(:,l+iterleft));
                    if any(A<trade) || A(j)<spacer
                        pop(Index_sorted_acc(l)).TradeOff...
                            =pop(Index_sorted_acc(l)).TradeOff+1;
                    end
                    iterleft=iterleft+1;
                end
            end
        end
    end
end
```

A.5 Adapted trade-off function

```
function pop=CalcTradeOff(pop,pop_prev,trade,spacer)

    M=numel(pop(1).Cost);
    n=numel(pop);
    n_prev=numel(pop_prev);

    % Initialisation
    for i=1:n
        pop(i).TradeOff=0;
        pop(i).Stop=false;
        pop(i).PrevIt=false;
    end
    for i=1:n_prev
        pop_prev(i).PrevIt=true;
    end
    pop_tot=[pop;pop_prev];
    [pop_tot, F_tot]=NonDominatedSorting(pop_tot);
    n_tot=numel(pop_tot);
    k=numel(F_tot);

    % Normalisation of objectives
    Cost=[pop_tot.Cost];
    CostSorted=zeros(M,n_tot);
    IndexSorted=zeros(M,n_tot);
    Index=1:n_tot; % the solutions of the previous iteration have indexes n+1 ...
        until n+n_prev
    Minima=zeros(M,1);
    Maxima=zeros(M,1);
    Range=zeros(M,1);
    for i=1:M
        [CostSorted(i,:),IndexSorted(i,:)]=sort(Cost(i,:));
        Minima(i)=CostSorted(i,1);
        Maxima(i)=CostSorted(i,end);
        Range(i)=abs(Maxima(i)-Minima(i));
    end
    for i=1:M
        for j=1:n_tot
            Cost(i,j)=(Cost(i,j)-Minima(i))/Range(i);
        end
    end

    % Trade-off calculation
    for i=1:k % for all non-dominated fronts
        Front_Cost=Cost(:,F_tot{i}); % select needed costs
        Front_Index=Index(F_tot{i}); % repmat with indexes in pop of selected ...
            solutions
        for j=1:M
            [¬,Sorter]=sort(Front_Cost(j,:));
            Index_sorted_acc=Front_Index(Sorter);
            Front_cost_sorted_obj=Front_Cost(:,Sorter);
            % for a solution in that front
            k_front=numel(F_tot{i});
            for l=2:k_front-1
                iterleft=1;
                iterright=1;
                if pop_tot(Index_sorted_acc(l)).PrevIt==false
                    while l+iterright<k_front
```

```
                        A=abs(Front_cost_sorted_obj(:,l)...
                        -Front_cost_sorted_obj(:,l+iterright));
                        if any(A<trade) || A(j)<spacer
                            if pop_tot(Index_sorted_acc(l+iterright)).PrevIt==false
                                pop_tot(Index_sorted_acc(l)).TradeOff...
                                =pop_tot(Index_sorted_acc(l)).TradeOff+1;
                            else
                                pop_tot(Index_sorted_acc(l)).Stop=true;
                            end
                        end
                        iterright=iterright+1;
                    end
                    while l-iterleft>0
                        A=abs(Front_cost_sorted_obj(:,l)...
                        -Front_cost_sorted_obj(:,l-iterleft));
                        if any(A<trade) || A(j)<spacer
                            if pop_tot(Index_sorted_acc(l-iterleft)).PrevIt==false
                                pop_tot(Index_sorted_acc(l)).TradeOff...
                                =pop_tot(Index_sorted_acc(l)).TradeOff+1;
                            else
                                pop_tot(Index_sorted_acc(l)).Stop=true;
                            end
                        end
                        iterleft=iterleft+1;
                    end
                end
            end
        end
    end
    pop=Truncate(pop_tot,n);
end
```

Appendix B

Attachments Chapter 5

B.1 Complete constrained NSGA-II algorithm

Algorithm 8 Constrained NSGA-II (framework based on Kalami (2015))

Require: Objective functions (Obj), constraints (Constr), N, t_{max}, p_c, p_m, μ, σ

INITIALISATION

for i = 1 **to** N **do**

$pop(i) = \texttt{RandomSolution}(\text{Obj})$

$pop(i) = \texttt{FeasibilityTest1}(pop(i), \text{Constr})$

end for

$pop, \mathcal{F} = \texttt{NonDominatedSorting}(pop)$

$pop = \texttt{CalcCrowdingDistance}(pop, \mathcal{F})$

$pop = \texttt{SortPopulation}(pop)$

MAIN LOOP

for it = 1 **to** t_{max} **do**

while $logic$ **do**

$pop_m = \texttt{Mutate}(pop, p_m, \mu, \sigma)$

$logic = \texttt{FeasibilityTest2}(pop_m, \text{Constr})$

end while

while $logic$ **do**

$pop_c = \texttt{Crossover}(pop, p_c)$

$logic = \texttt{FeasibilityTest2}(pop_c, \text{Constr})$

end while

$pop_{it} = pop \cup pop_c \cup pop_m$

$pop_{it}, \mathcal{F}_{it} = \texttt{NonDominatedSorting}(pop_{it})$

$pop_{it} = \texttt{CalcCrowdingDistance}(pop_{it}, \mathcal{F}_{it})$

$pop_{it} = \texttt{SortPopulation}(pop_{it})$

$pop = pop_{it}(1:N)$

$pop, \mathcal{F} = \texttt{NonDominatedSorting}(pop)$

$pop = \texttt{CalcCrowdingDistance}(pop, \mathcal{F})$

$pop = \texttt{SortPopulation}(pop)$

end for

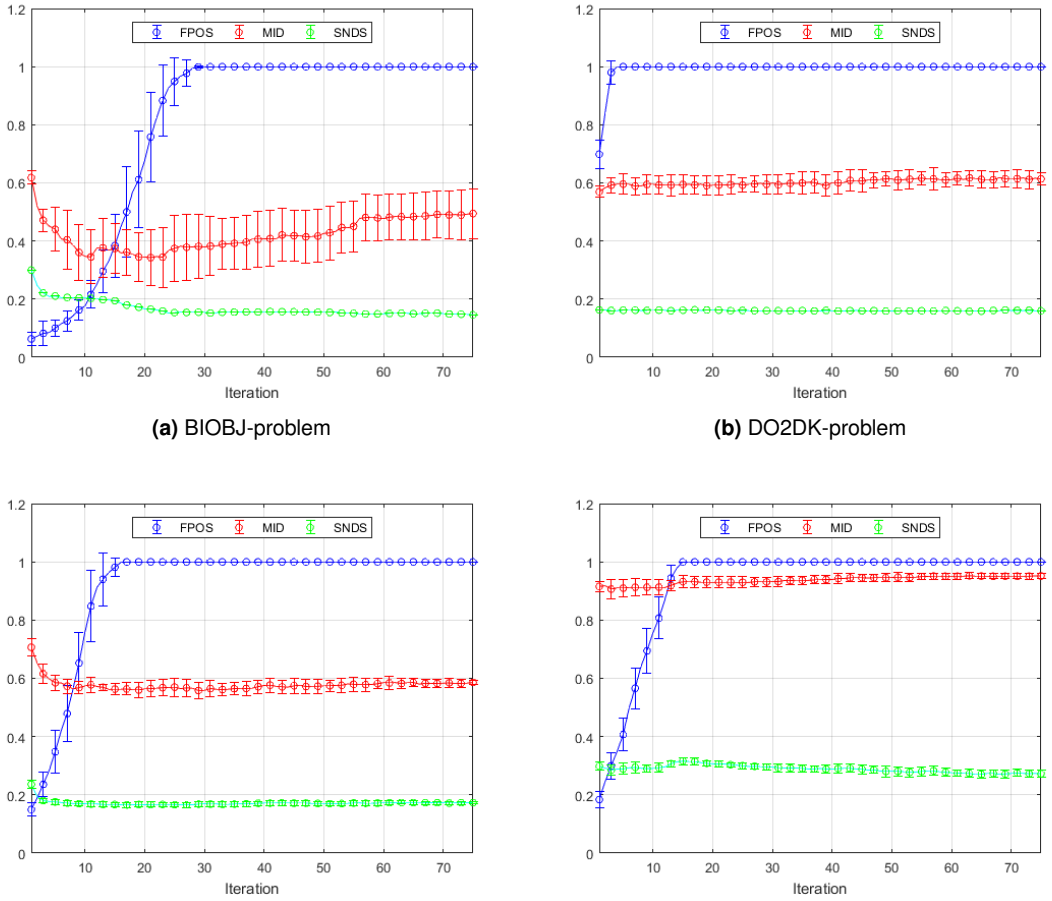
B.2 Complete constrained NSGA-III algorithm

Algorithm 9 Constrained NSGA-III (framework based on Kalami (2016))

Require: Objective functions (Obj), constraints (Constr), d, N, t_{max}, p_c, p_m, μ, σ

INITIALISATION
for i = 1 **to** N **do**
$pop(i) = \texttt{RandomSolution}(\text{Obj})$
$pop(i) = \texttt{FeasibilityTest1}(pop(i), \text{Constr})$
end for
$pop, \mathcal{F} = \texttt{NonDominatedSorting}(pop)$
$pop = \texttt{NormalisePopulation}(pop)$
$pop = \texttt{AssociateToRefPoint}(pop, \mathcal{F})$
$pop = \texttt{SortPopulation}(pop)$
MAIN LOOP
for it = 1 **to** t_{max} **do**
while $logic$ **do**
$pop_m = \texttt{Mutate}(pop, p_m, \mu, \sigma)$
$logic = \texttt{FeasibilityTest2}(pop_m, \text{Constr})$
end while
while $logic$ **do**
$pop_c = \texttt{Crossover}(pop, p_c)$
$logic = \texttt{FeasibilityTest2}(pop_c, \text{Constr})$
end while
$pop_{it} = pop \cup pop_c \cup pop_m$
$pop_{it}, \mathcal{F}_{it} = \texttt{NonDominatedSorting}(pop_{it})$
$pop_{it} = \texttt{NormalisePopulation}(pop_{it})$
$pop_{it} = \texttt{AssociateToRefPoint}pop_{it}, \mathcal{F})$
$pop_{it} = \texttt{SortPopulation}(pop_{it})$
$pop = pop_{it}(1:N)$
$pop, \mathcal{F} = \texttt{NonDominatedSorting}(pop)$
$pop = \texttt{NormalisePopulation}(pop)$
$pop = \texttt{AssociateToRefPoint}(pop, \mathcal{F})$
$pop = \texttt{SortPopulation}(pop)$
end for

B.3 Case studies: Performance parameters



(a) BIOBJ-problem

(b) DO2DK-problem

(c) CONSTR-problem

(d) TNK-problem

Figure B.1: Performance plots of the constrained NSGA-III algorithm in case of the bi-objective case studies (average taken over 10 repetitions).

Appendix C

Attachments Chapter 6

C.1 Complete tM-NSGA-II algorithm

Algorithm 10 tM-NSGA-II (framework based on Kalami (2015))

Require: Objective functions (Obj), constraints (Constr), N, t_{max}, p_c, p_m, μ, σ, Δt, Δr, ΔMID

```
for i = 1 to M do
    pop(i) = AnchorPoints(Obj,Constr)
end for
for i = 1 to N − M do
    pop(i) = RandomSolution(Obj)
    pop(i) = FeasibilityTest1(pop(i), Constr)
end for
for it = 1 to t_max do
    while logic do
        pop_m = Mutate(pop, p_m, μ, σ)
        logic = FeasibilityTest2(pop_m, Constr)
    end while
    while logic do
        pop_c = Crossover(pop, p_c)
        logic = FeasibilityTest2(pop_c, Constr)
    end while
    pop_it = pop ∪ pop_c ∪ pop_m
    pop_it, F_it = NonDominatedSorting(pop_it)
    pop_it = CalcCrowdingDistance(pop_it, F_it)
    pop_it = CalcTradeOff(pop_it, F_it, Δt, Δr)
    pop_it = tM − SortPopulation(pop_it)
    pop = pop_it(1 : N)
    pop, F = NonDominatedSorting(pop)
    MID_it = CalculateMID(pop)
    FPOS_it = |F_1|/N
    if |MID_it − MID_it−1| ≤ ΔMID and FPOS_it = 1 then
        break
    end if
end for
```

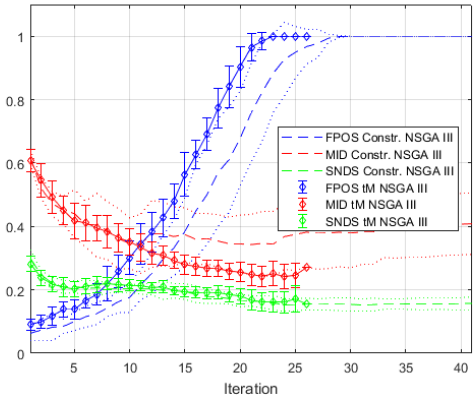
C.2 Complete tM-NSGA-III algorithm

Algorithm 11 tM-NSGA-III (framework based on Kalami (2016))

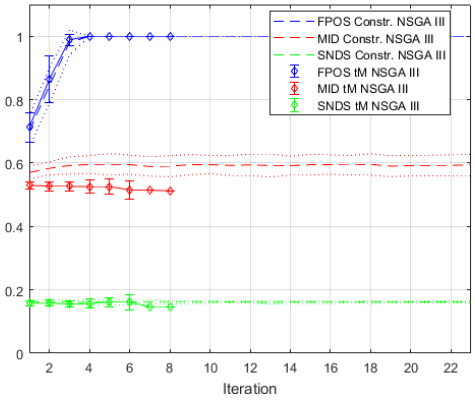
Require: Objective functions (Obj), constraints (Constr), d, N, t_{max}, p_c, p_m, μ, σ, Δt, Δr, ΔMID

for i = 1 **to** M **do**
  $pop(i) = \texttt{AnchorPoints}(\text{Obj},\text{Constr})$
end for
for i = 1 **to** $N-M$ **do**
  $pop(i) = \texttt{RandomSolution}(\text{Obj})$
  $pop(i) = \texttt{FeasibilityTest1}(pop(i), \text{Constr})$
end for
for it = 1 **to** t_{max} **do**
  while $logic$ **do**
    $pop_m = \texttt{Mutate}(pop, p_m, \mu, \sigma)$
    $logic = \texttt{FeasibilityTest2}(pop_m, \text{Constr})$
  end while
  while $logic$ **do**
    $pop_c = \texttt{Crossover}(pop, p_c)$
    $logic = \texttt{FeasibilityTest2}(pop_c, \text{Constr})$
  end while
  $pop_{it} = pop \cup pop_c \cup pop_m$
  $pop_{it}, \mathcal{F}_{it} = \texttt{NonDominatedSorting}(pop_{it})$
  $pop_{it} = \texttt{NormalisePopulation}(pop_{it})$
  $pop_{it} = \texttt{AssociateToRefPoint}pop_{it}, \mathcal{F})$
  $pop_{it} = \texttt{CalcTradeOff}(pop_{it}, \mathcal{F}_{it}, \Delta t, \Delta r)$
  $pop_{it} = \texttt{tM}-\texttt{SortPopulation}(pop_{it})$
  $pop = pop_{it}(1:N)$
  $pop, \mathcal{F} = \texttt{NonDominatedSorting}(pop)$
  $MID_{it} = \texttt{CalculateMID}(pop)$
  $FPOS_{it} = |\mathcal{F}_1|/N$
  if $|MID_{it} - MID_{it-1}| \leq \Delta MID$ **and** $FPOS_{it} = 1$ **then**
    `break`
  end if
end for

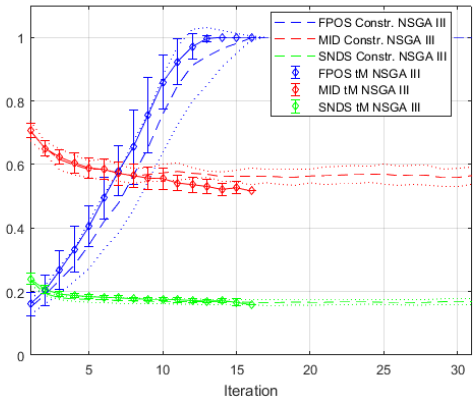
C.3 Case studies: Performance parameters



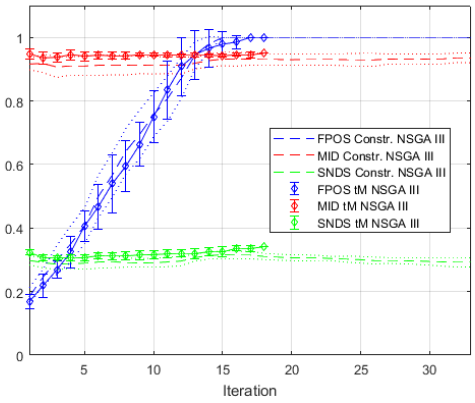
(a) BIOBJ-problem



(b) DO2DK-problem



(c) CONSTR-problem



(d) TNK-problem

Figure C.1: Performance plots of the tM-NSGA-III algorithm in case of the bi-objective case studies (average taken over 10 repetitions). The iteration axis is cropped.

Appendix D

Attachments Chapter 7

D.1 Complete tDOM-NSGA-II algorithm

Algorithm 12 tDOM-NSGA-II (framework based on Kalami (2015))

Require: Objective functions (Obj), constraints (Constr), N, t_{max}, p_c, p_m, μ, σ, Δt, Δr

for i = 1 **to** M **do**
  $pop(i) = \texttt{AnchorPoints}(\text{Obj,Constr})$
end for
for i = 1 **to** $N - M$ **do**
  $pop(i) = \texttt{RandomSolution}(\text{Obj})$
  $pop(i) = \texttt{FeasibilityTest1}(pop(i), \text{Constr})$
end for
for it = 1 **to** t_{max} **do**
  while $logic$ **do**
    $pop_m = \texttt{Mutate}(pop, p_m, \mu, \sigma)$
    $logic = \texttt{FeasibilityTest2}(pop_m, \text{Constr})$
  end while
  while $logic$ **do**
    $pop_c = \texttt{Crossover}(pop, p_c)$
    $logic = \texttt{FeasibilityTest2}(pop_c, \text{Constr})$
  end while
  $pop_{it} = pop \cup pop_c \cup pop_m$
  $pop_{it}, \mathcal{F}_{it} = \texttt{NonDominatedSorting}(pop_{it})$
  $pop_{it} = \texttt{CalcCrowdingDistance}(pop_{it}, \mathcal{F}_{it})$
  $pop_{it} = \texttt{AdaptedTradeOff}(pop_{it}, pop_{it-1}, \Delta t, \Delta r)$
  $pop_{it} = \texttt{tDOM} - \texttt{SortPopulation}(pop_{it})$
  $pop = pop_{it}(1:N)$
  $pop, \mathcal{F} = \texttt{NonDominatedSorting}(pop)$
  $FPOS_{it} = |\mathcal{F}_1|/N$
  if all pop_{it}.Stop $= true$ **and** $FPOS_{it} = 1$ **then**
    break
  end if
end for

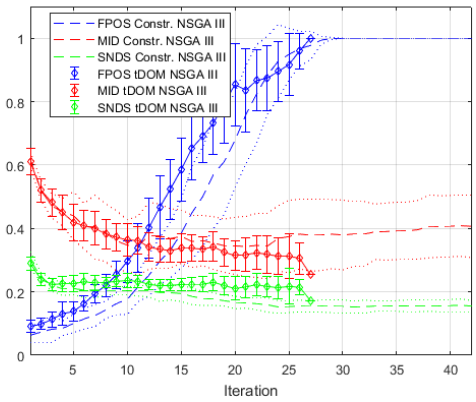
D.2 Complete tDOM-NSGA-III algorithm

Algorithm 13 tDOM-NSGA-III (framework based on Kalami (2016))

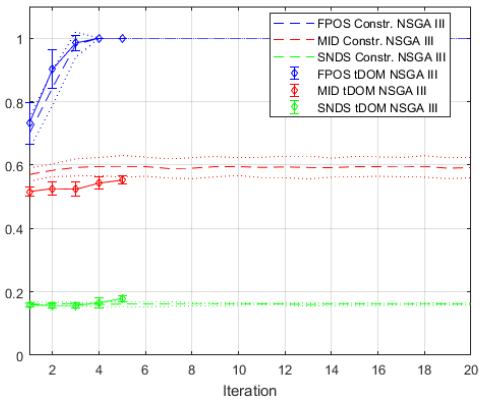
Require: Objective functions (Obj), constraints (Constr), d, N, t_{max}, p_c, p_m, μ, σ, Δt, Δr

for i = 1 **to** M **do**
  $pop(i) = \texttt{AnchorPoints}(\text{Obj},\text{Constr})$
end for
for i = 1 **to** $N - M$ **do**
  $pop(i) = \texttt{RandomSolution}(\text{Obj})$
  $pop(i) = \texttt{FeasibilityTest1}(pop(i), \text{Constr})$
end for
for it = 1 **to** t_{max} **do**
  while $logic$ **do**
    $pop_m = \texttt{Mutate}(pop, p_m, \mu, \sigma)$
    $logic = \texttt{FeasibilityTest2}(pop_m, \text{Constr})$
  end while
  while $logic$ **do**
    $pop_c = \texttt{Crossover}(pop, p_c)$
    $logic = \texttt{FeasibilityTest2}(pop_c, \text{Constr})$
  end while
  $pop_{it} = pop \cup pop_c \cup pop_m$
  $pop_{it}, \mathcal{F}_{it} = \texttt{NonDominatedSorting}(pop_{it})$
  $pop_{it} = \texttt{NormalisePopulation}(pop_{it})$
  $pop_{it} = \texttt{AssociateToRefPoint}pop_{it}, \mathcal{F})$
  $pop_{it} = \texttt{AdaptedTradeOff}(pop_{it}, pop_{it-1}, \Delta t, \Delta r)$
  $pop_{it} = \texttt{tDOM} - \texttt{SortPopulation}(pop_{it})$
  $pop = pop_{it}(1 : N)$
  $pop, \mathcal{F} = \texttt{NonDominatedSorting}(pop)$
  $FPOS_{it} = |\mathcal{F}_1|/N$
  if all $pop_{it}.\texttt{Stop} = true$ **and** $FPOS_{it} = 1$ **then**
    `break`
  end if
end for

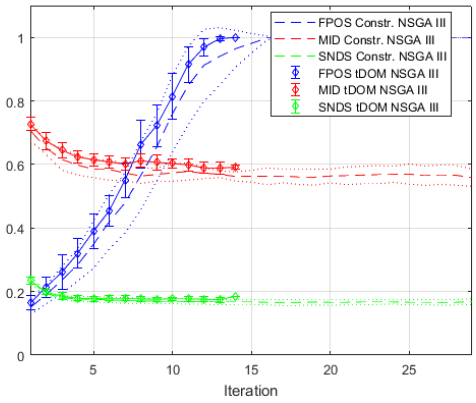
D.3 Case studies: Performance parameters



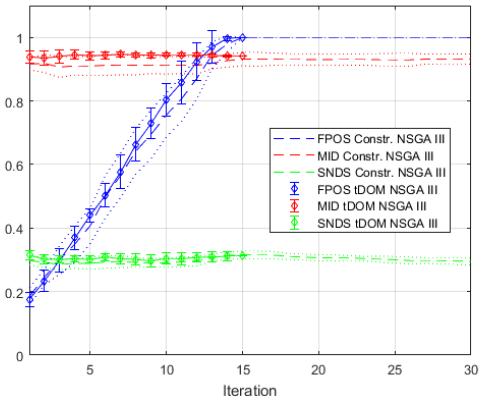
(a) BIOBJ-problem



(b) DO2DK-problem



(c) CONSTR-problem



(d) TNK-problem

Figure D.1: Performance plots of the tDOM-NSGA-III algorithm in case of the bi-objective case studies (average taken over 10 repetitions). The iteration axis is cropped.

FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
TECHNOLOGIECAMPUS GENT
Gebroeders De Smetstraat 1
8200 GENT, België
tel. + 32 50 66 48 00
iiw.gent@kuleuven.be
www.iiw.kuleuven.be