Parallel Processing for Dynamic Multi-objective Optimization

Mario Cámara Sola

DISSERTATION SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Supervisors: **Dr. Julio Ortega Lopera**
Dr. Francisco de Toro Negro

Dept. of Computer Architecture and Computer Technology
GRANADA, APRIL 2010

Editor: Editorial de la Universidad de Granada
Autor: Mario Cámara Sola
D.L.: GR 3232-2010
ISBN: 978-84-693-4423-1

D. Julio Ortega Lopera, catedrático de la Universidad de Granada, y D. Francisco Jesús de Toro Negro, profesor contratado doctor de la Universidad de Granada

CERTIFICAN

Que la memoria titulada

Parallel Processing for Dynamic Multi-objective Optimization

ha sido realizada por D. Mario Cámara Sola bajo nuestra dirección en el departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada, para optar al grado de Doctor.

En Granada, a 6 de abril de 2010.

Fdo. Julio Ortega Lopera　　　Fdo. Francisco J. de Toro Negro

Procesamiento Paralelo para Optimización Dinámica Multiobjetivo

Memoria presentada por

Mario Cámara Sola

Para optar al grado de

Doctor en Informática con Mención Europea

por la Universidad de Granada

Fdo. Mario Cámara Sola

____

Посвящается моей любимой Юле за её
понимание, терпение и поддержку

“The real voyage of discovery consists not in
seeking new landscapes but in having new eyes”
Marcel Proust

Учиться, учиться и ещё раз учиться
В. И. Ленин

“Learn, learn and once more learn”
V. I. Lenin

“The greatest challenge to any thinker is stating the
problem in a way that will allow a solution.”
Bertrand Russell

Acknowledgements

It is common belief that embarking upon a PhD is a lonely task where the PhD candidate will have to carry on a wide range of activities but most of them without external help.

Hopefully, my personal experience when I am finishing this PhD thesis is that I have not been alone in any moment throughout this long way.

Julio Ortega and **Francisco de Toro**, my two supervisors, were always there: ready and zealous but, above all, they were always willing to point me out the next step to follow in my research. To them my warmest thanks for spending and sharing their time with me during these years.

Moreover, this PhD could not have come to life without the support of the Computer Architecture and Technology Department and its staff, among them **Alberto Prieto**, **Manuel Rodríguez**, **JJ Merelo**, **José Luis Bernier**, **Ignacio Rojas**, **Francisco Illeras** and **Encarnación Redondo** deserve special mention.

I was also very lucky to meet very wonderful workmates and I want to thank you all for your interesting views on all kind of issues that opened my eyes many times: **Pablo Cascón**, **José Miguel Urquiza**, **Juan Luis Jiménez**, **Pablo García**, **Ginés Rubio**, **Antonio Mora**, **Luis Javier Herrera**, **Alberto Guillén** and **Richard Carrillo**.

I am very grateful to my whole family for supporting me in this long journey, and specially to **Yulia**, who suffered my absolute lack of free time to spend with her during the last nine months.

Last but not least I am very grateful to the excellent researchers who agreed to host me in their research labs in order to learn from them and their colleagues: **Ben Paechter** at Napier University (Edinburgh) and **Eckart Zitzler** at Zurich ETH.

This PhD has been developed with the economic support provided by a research fellowship given by the Andalusian Regional Ministry of

Innovation, Science and Enterprise (Consejería de Innovación, Ciencia y Empresa), currently Regional Ministry of Economy, Innovation and Science (Consejería de Economía, Innovación y Ciencia) of the Junta de Andalucía. Moreover, my research visit to Edinburgh was funded by the HPC-Europa2 programme.

Contents

List of Abbreviations

List of abbreviations used in this thesis

- OP - (Single-objective) Optimization Problem
- MOP - Multi-objective Optimization Problem
- DOP - Dynamic Optimization Problem
- DMO - Dynamic Multi-objective Optimization
- EA - Evolutionary Algorithm
- MOEA - Multi-objective Evolutionary Algorithm
- GP - Genetic Programming
- NFL - No Free Lunch theorems
- TSP - Traveling Salesman Problem
- pdMOEA - Parallel Dynamic MOEA
- pdMOEA+ - Improved Parallel Dynamic MOEA

Abstract

The main objective of this PhD thesis is to advance the field of parallel multi-objective evolutionary algorithms to solve dynamic multi-objective optimization problems. Thus, the research presented in this thesis involves three different, although related, fields:

- Multi-objective evolutionary algorithms (MOEA),
- Dynamic multi-objective optimization (DMO) problems, and
- Parallelization of MOEAs to solve DMO problems.

The degree of advancement of the research varies for each of the afore-mentioned topics, from a full-fledged research field as it is the MOEA topic to a new emerging subject as it happens with dynamic multi-objective optimization.

Nevertheless, proposals to improve further the three afore-mentioned subjects have been made in this thesis.

First of all, this thesis introduces a *low-cost* MOEA able to deal with multi-objective problems within more restrictive time limits than other state-of-the-art can do.

Secondly, the field of dynamic optimization is reviewed and some additions are made so that the field moves forward to tackle dynamic multi-objective problems. This has been facilitated by the introduction of performance measures for problems that are both dynamic and multi-objective. Moreover, modifications are proposed for two of the five *de facto* standard test cases for DMO problems.

Thirdly, the parallelization of MOEAs to solve DMO problems is addressed with two different proposed approaches:

- A hybrid master-worker and island approach called pdMOEA, and

- A fully distributed approach called pdMOEA+.

These two approaches are compared side-by-side with the test cases already mentioned.

Finally, future work to follow upon the achievements of this thesis is outlined.

"La verdadera ciencia enseña, por encima de todo, a dudar y a ser ignorante."

Miguel de Unamuno

Resumen

ESTA memoria representa el resultado de más de cuatro años de investigación llevada a cabo en el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada.

El objetivo principal de esta tesis doctoral es producir un avance en el campo de los algoritmos evolutivos multiobjetivo y paralelos para resolver problemas dinámicos multiobjetivo. Para ello, la investigación presentada en esta tesis trata con tres campos que, aunque diferentes, están relacionados entre sí:

- Algoritmos evolutivos de optimización multiobjetivo (MOEA, por sus siglas en inglés)
- Optimización dinámica multiobjetivo (DMO, por sus siglas en inglés), y
- La paralelización de algoritmos que resuelven problemas DMO con el uso de MOEAs.

Estos tres temas se entremezclan en toda la tesis para dar como resultado un enfoque unificado que sea capaz de afrontar problemas DMO mediante el uso de MOEAs.

El grado de desarrollo de la investigación varía para cada uno de los tópicos mencionados con anterioridad, desde un campo de investigación muy bien nutrido como es el tema de los algoritmos multiobjetivo hasta un tema nuevo y emergente como la optimización dinámica multiobjetivo.

No obstante lo anterior, en esta tesis se incluyen propuestas para desarrollar aún más las tres materias ya citadas.

Primero, se introduce un algoritmo evolutivo multiobjetivo de bajo coste que es capaz de encarar problemas multiobjetivo con limitaciones de tiempo más restrictivas que las que pueden afrontar otros algoritmos punteros.

Segundo, se revisa el campo de la optimización dinámica y se realizan algunos añadidos para que el campo evolucione hacía la posibilidad de resolver problemas dinámicos multiobjetivo. Esto se basa en la introducción de medidas de rendimiento para problemas que son, a la vez, dinámicos y multiobjetivo. Además, se proponen modificaciones para dos de los cinco casos de prueba estándar para problemas de optimización dinámica multiobjetivo.

En tercer lugar, se afronta la paralelización de algoritmos evolutivos multiobjetivo para resolver problemas dinámicos multiobjetivo con dos enfoques diferentes propuestos:

- Un enfoque híbrido entre maestro-trabajador e isla que se ha llamado pdMOEA, y
- Un enfoque completamente distribuido que ha sido llamado pdMOEA+.

Estos dos enfoques son comparados exhaustivamente con los casos de prueba ya mencionados.

Objetivos

Como se ha dicho en el epigrafe anterior, el objetivo principal de esta tesis es ofrecer *un enfoque paralelo fiable, preciso, rápido y escalable que sea capaz de tratar problemas de optimización dinámica multiobjetivo bajo restricciones de tiempo.*

La idea subyacente en este objetivo es poder resolver aquellos problemas que son a la vez dinámicos y multiobjetivo mediante el uso de procedimientos paralelos escalables. De tal forma que si un problema crece añadiéndole nuevas instancias, se pueda resolver requiriendo el mismo tiempo que al comienzo mediante la adición de más procesadores al entorno de computación.

Esto podría suceder en entornos reales (Gupta and Sivakumar, 2006; Lee, Teng, Chew, Karimi, Lye, Lendermann, Chen, and Koh, 2005) tales como una fábrica donde un proceso de fabricación conlleva seis máquinas, y nuestro algoritmo es usado para planificar las diferentes tareas que hay que acabar cada hora. Si la fábrica es ampliada con cuatro máquinas nuevas, el algoritmo debe poder mantener los límites temporales antiguos mientras permite planificar las diez máquinas en lugar de solo seis. Esto demandaría al algoritmo ejecutarse de una manera más rápida y, en caso de un entorno y algoritmo secuenciales, solo podría conseguirse esto cambiando el computador actual por alguno otro mucho más caro.

No obstante, si se usara un enfoque paralelo, como un sistema *cluster*, el algoritmo podría mantener los límites de tiempo anteriores si se ejecutara en unos cuantos procesadores más. Este enfoque se ha demostrado ser más económico y confiable porque es más fácil y barato añadir un nuevo procesador a un entorno paralelo que encontrar un procesador más potente para reemplazar otro en un entorno secuencial. Esto se ha hecho más patente aún con la proliferación de las soluciones ya listas de chips con múltiples núcleos.

Como se ha dicho antes, para diseñar dicho algoritmo paralelo es necesario trabajar en colaboración con las áreas de investigación anteriormente citadas porque todas ellas tienen algún impacto en el resultado final. Además, el nivel de desarrollo de cualquiera de estas áreas varía, lo que complica el alcanzar el objetivo de esta tesis. Esto se traduce en que algunas de estas áreas requerían algún desarrollo más, que se ha realizado en esta tesis.

El campo de los algoritmos MOEA es el área más desarrollada de las tres. En ella, los investigadores pueden encontrar múltiples algoritmos que mejoran y solventan las debilidades que presentaban los algoritmos que se habían propuesto con anterioridad. Teniendo esto en cuenta, esta tesis no intenta dar otro algoritmo multiobjetivo más que entre a rivalizar con los algoritmos que ya se encuentran disponibles. En su lugar, en esta tesis se describe un nuevo algoritmo MOEA que es muy rápido pero que ha sido diseñado con el claro objeto de proporcionar un algoritmo de *bajo coste* que se comporte mucho más rápido que los algoritmos existentes.

En esta tesis, la locución algoritmo de bajo coste proviene de las líneas aéreas de bajo coste y de su modelo de negocio comparado con el de las líneas aéreas regulares. Esto es así porque el algoritmo propuesto es capaz de dar soluciones cuya calidad es prácticamente la misma que la calidad de las soluciones obtenidas con los algoritmos más punteros como SPEA2 y NSGA-II. Este algoritmo propuesto emula la idea de los vuelos de bajo coste que permiten volar a aeropuertos que están localizados casi tan cercanos del centro de las ciudades como los aeropuertos utilizados por las líneas regulares. Pero al igual que sucede con los vuelos de bajo coste, el algoritmo aquí propuesto proporciona estas soluciones de una manera que es más eficiente en términos de tiempo que la manera en que lo hacen los algoritmos tradicionales.

Si un vuelo de bajo coste ofrece una reducción en el precio del billete de avión a cambio de aumentar la distancia hasta el centro de la ciudad, el algoritmo de bajo coste mejora el número de soluciones que produce por unidad de tiempo a costa de la distancia de esas soluciones al frente real de Pareto. Lógicamente, este compromiso en la distancia al frente de Pareto se hace de tal forma que se mantengan dentro de un cierto nivel de calidad mínimo.

El interés de tener un algoritmo MOEA que proporciona más soluciones por unidad de tiempo mientras que no compromete demasiado la calidad de las soluciones recae en el hecho de que tales algoritmos serán más adecuados para resolver problemas DMO dentro de límites temporales restrictivos, y así, adaptarse a nuevas restricciones de tiempo más pequeñas.

Una vez que se tiene un algoritmo MOEA o un conjunto de ellos, es importante proporcionar un entorno adecuado para resolver problemas dinámicos. Este entorno debería estar compuesto por teoría y práctica. Desafortunadamente, en el caso de los problemas dinámicos, no ha habido prácticamente desarrollos para los problemas dinámicos multiobjetivo. Debido a esto, se debe dar un armazón teórico para tratar los problemas DMO.

Este entorno debería dar medidas de rendimiento y criterios para este tipo de problemas que son dinámicos y multiobjetivo a la vez. Además, debería proporcionar casos de prueba que pudieran ser usados por los investigadores en el campo. Es esencial que este entorno teórico pueda ser usado no ya para casos de prueba simples sino también para problemas DMO tomados de situaciones del mundo real.

Por último, se debería dar una paralelización de estos algoritmos MOEA para la resolución de problemas DMO. Este enfoque paralelo debería, al menos, mantener la calidad de las soluciones dadas por los algoritmos secuenciales, a la vez que redujera el tiempo de ejecución requerido. También debería mostrar que es escalable, para permitir que se añadieran más nodos para afrontar problemas con instancias más grandes.

De forma adicional, este esquema paralelo debería implementar cualidades deseadas como la habilidad de ejecutar diferentes algoritmos MOEA, ser completamente independiente del problema y no depender de parámetros introducidos por el usuario.

De forma resumida, los objetivos que persigue esta tesis son:

- Ofrecer un análisis y un repaso de las medidas de rendimiento disponibles para problemas DMO, con especial interés en aquellas destinadas a la parte dinámica de estos problemas.

- Extender el armazón teórico para la optimización dinámica multiobjetivo añadiendo medidas de rendimiento para los casos que ya estuvieran cubiertos y establecer las bases para futuros estudios en dicha área.

- Implementar un algoritmo MOEA de bajo tiempo de ejecución que sea adecuado para paralelizar problemas DMO.

- Incrementar el número de soluciones por unidad de tiempo producidas por el algoritmo anterior.

- Realizar una comparación de este nuevo algoritmo MOEA con otros algoritmos ampliamente conocidos.

- Proponer un procedimiento paralelo capaz de resolver problemas DMO con buenas ganancias en velocidad y que permita ejecutar diferentes algoritmos MOEA. Este procedimiento deberá ser capaz de resolver problemas DMO con restricciones de tiempo sin comprometer la calidad de las soluciones obtenidas.

- Estudiar otras mejoras realizables a este enfoque paralelo.

- Comparar el procedimiento paralelo propuesto con el enfoque secuencial a través de resultados experimentales.

- Recopilar las preguntas que surjan como consecuencia de esta tesis y que deberían ser respondidas en investigaciones futuras.

Estructura

Después de esta introducción, la estructura de esta tesis se describe a continuación:

- El **Capítulo 1. Introduction and Field Review** se dedica a la revisión de los tres temas principales que dan forma a esta tesis antes de entrar en detalles en ellos en el resto de los capítulos.

- El **Capítulo 2. Performance Evaluation** trata sobre medidas de rendimiento para problemas de optimización multiobjetivo dinámica. Primero comienza con una recopilación de las medidas propuestas para problemas dinámicos de un solo objetivo y para problemas multiobjetivo estacionarios. Después, se proponen algunas medidas para problemas que son dinámicos y multiobjetivo. Por último, se introducen los casos de prueba usados para problemas DMO junto con algunas modificaciones que se proponen.

- El **Capítulo 3. SFGA2: An improved version of SFGA** comienza con una revisión detallada de tres algoritmos MOEA usados actualmente: SFGA, SPEA2 y NSGA-II. A continuación, se describe profusamente el algoritmo *Single Front Genetic Algorithm 2* (SFGA2) y se da una justificación para su uso. Finalmente, se realiza una comparación en términos de calidad de las soluciones y tiempo de ejecución de estos cuatro algoritmos MOEA: SFGA, SFGA2, SPEA2 y NSGA-II.

- El **Capítulo 4. Parallel Procedures for DMO** explora las posibles maneras que disponen los investigadores para producir enfoques paralelos para resolver problemas dinámicos multiobjetivo mediante el uso de algoritmos MOEA. En esta tesis se proponen dos procedimientos para paralelizar algoritmos MOEA para resolver problemas DMO. El primero, llamado pdMOEA, es un enfoque híbrido que puede variar entre un paradigma maestro-trabajador a un paradigma de islas. Un segundo procedimiento completamente distribuido se propone con el fin de mejorar más aún el enfoque híbrido. Una vez se describen dichos procedimientos se dan resultados experimentales para los dos con el fin de evaluar el rendimiento de ambos enfoques paralelos.

- El **Capítulo 5. Summary Conclusions and Contributions** es el último capítulo y ofrece un resumen de las conclusiones de esta tesis además de recoger las principales aportaciones hechas y las publicaciones que se han realizado.

- **Conclusiones Finales y Aportaciones** es un apéndice redactado en español que resume las conclusiones y principales aportaciones que han surgido de esta tesis. Incluye también las publicaciones que se han realizado y las líneas de trabajo futuro que se pueden seguir a partir de esta tesis.

Publicaciones

El trabajo realizado en esta tesis doctoral se ha materializado en el siguiente número de publicaciones: dos revistas internacionales, un capítulo de libro por invitación, cinco congresos internacionales, tres congresos nacionales y dos de otro tipo.

"The wise man doesn't give the right answers, he poses the right questions."

Claude Levi-Strauss

Preface

THIS dissertation is the result of more than four years of research held at the Department of Computer Architecture and Computer Technology of the University of Granada. The research that is presented in this thesis has been focused on three main topics:

- Multi-objective optimization evolutionary algorithms (MOEA),
- Dynamic multi-objective optimization (DMO), and
- The parallelization of algorithms that solve DMO problems by using MOEAs

These three topics intersperse along all the thesis to provide a unified approach that is able to tackle DMO problems by means of parallel MOEAs.

Objectives

The main goal of this thesis is to provide *a reliable, accurate, fast and scalable parallel approach able to deal with dynamic multi-objective optimization problems within time constraints*.

The idea behind this objective is to be able to solve problems that are both dynamic and multi-objective by means of scalable parallel procedures so that if a problem becomes bigger by the addition of new instances, it can be solved requiring the same time as before by adding more processors to the computing environment.

This could happen in a real world setting (Gupta and Sivakumar, 2006; Lee et al., 2005) such as a factory where a manufacturing process involves six machines, and our algorithm is used to schedule the different tasks to be done on a hourly basis. If the factory is expanded with for new machines, the algorithm must be able to keep up with the older time limits but providing schedules for ten machines instead of only six. This would demand the algorithm to run faster and, given a sequential environment and a sequential algorithm, this could only be provided by changing the underlying computer by another much more expensive computer.

But if a parallel approach is used, such as a cluster system, the algorithm will be able to keep up with the older time limits by running in a few more processors. This approach has been shown to be more cost-effective and reliable because it is easier and cheaper to add a new processor to a parallel environment than to find a faster processor to replace another one in a sequential environment. This has become even clearer with the proliferation of off-the-shelf multi-core solutions.

As it has been said before, in order to come up with such a parallel algorithm it is necessary to work with the afore-mentioned research areas because all of them have an impact on the final result. In addition, the level of development of any of these areas varies, which is a complication to reach the goal of this thesis. This means that some further development was needed, and consequently done on this thesis, on some of the areas.

The MOEA field is the more developed area of the three ones. In it, researchers can find multiple algorithms that improve upon the weaknesses of the previously proposed ones. Bearing this in mind, this thesis does not try to provide just another MOEA to enter in direct competition with those algorithms that are already available. Nonetheless, in this thesis a new very fast MOEA is given but with a clear aim of providing a *low-cost* algorithm that behaves much faster than the available algorithms.

In this thesis, the term low-cost algorithm comes from the low-cost airlines and from their business model in comparison with the regular airlines. This is because, it is provided a MOEA that is able to produce solutions whose quality is almost the same as the quality of the solutions provided by state-of-the-art algorithms such as SPEA2 and NSGA-II. This proposed algorithm mimics the idea of low-cost flights that allow the traveller to fly to airports which are located almost so near to the city centres as the airports to which regular airlines fly. But as it happens with low-cost flights, our proposed algorithm provides these solutions in a way that is more time efficient than the way standard algorithms do.

If a low-cost flight offers a reduction in the price of a plane ticket on the expense of the distance to a city centre, a low-cost algorithm improves in the number of solutions it produces per time unit, on the expense of the distance of those solutions to the real Pareto front. Logically, this compromise on the distance to the Pareto front keeps the solutions within some minimum of acceptable quality.

The interest of having a MOEA that provides more solutions per time unit while not compromising too much the quality of the solutions relies on the fact that such algorithms will be more adequate to solve DMO problems within restrictive time limits, and in turn, to adapt seamlessly to new shorter time restrictions.

Once a MOEA or a set of them is available, it is important to provide an adequate framework to solve dynamic problems. This framework should be comprised of theoretical and practical issues. Unfortunately, in the case of dynamic problems, there were almost not development for dynamic and multi-objective problems. Due to this, it must be provided a theoretical framework to deal with DMO problems.

This framework should give performance measures and criteria for these kind of problems that are dynamic and multi-objective at the same time. In addition, it should provide test cases to be used by the researchers on the field. It is essential that this theoretical framework can be used not only for toy test cases but also for real-world DMO problems.

Finally, a parallelization of these MOEAs should be provided in order to solve DMO problems. This parallel approach should, at least, maintain the quality of the solutions provided by a sequential approach, while decreasing the required running time. It should also show that it is scalable in order to allow more nodes to be added when dealing with bigger instances of the problems.

Additionally, this parallel scheme should implement desired features such as the ability to run different MOEAs, to be fully independent of the problem and not to depend on many user parameters.

In summary, the objectives that this thesis pursues are:

- To provide an analysis and review of the available performance measures for DMO, specifically aimed at the dynamic part of these problems.

- To extend the theoretical framework for DMO by adding performance measures for cases that were not covered and to establish the grounds for further study in this area.
- To implement a low running-time MOEA that is suitable to parallelize DMO problems.
- To increase the number of solutions per time unit produced by the earlier algorithm.
- To provide a comparison of this new MOEA with other widely known MOEAs.
- To propose a parallel procedure able to solve DMO problems with good speedups and that allows to run different MOEAs. This procedure must be able to solve DMO problems within time restrictions without compromising the quality of the given solutions.
- To study further improvements to this parallel procedure.
- To compare the proposed parallel procedures with the sequential approach through experimental results.
- To list the questions that have arisen from this thesis and that should be addressed in further research.

Structure

After this introduction, the structure of this thesis is described in what follows:

- **Chapter 1. Introduction and Field Review** is devoted to review the three main topics that give form to this thesis before delving deeply into them in the rest of the chapters.

- **Chapter 2. Performance Evaluation** deals with performance measures for dynamic multi-objective optimization problems. First, it begins with a compilation of proposed measures for single-objective dynamic problems and for stationary multi-objective problems. Afterwards, some measures are proposed for problems that are both dynamic and multi-objective. Finally, the test cases used for DMO problems are introduced and some modifications for them are proposed.

- **Chapter 3. SFGA2: An improved version of SFGA** begins with a detailed review of three currently used MOEAs: SPEA2, NSGA-II and SFGA. Then, the *Single Front Genetic Algorithm 2* (SFGA2) is thoroughly described with a justification for its introduction. Lastly, a comparison with respect to quality of the solutions and execution time of these four MOEAs, SFGA, SFGA2, NSGA-II and SPEA2, is done.

- **Chapter 4. Parallel Procedures for DMO** explores the possible ways that researchers have to produce parallel approaches to solve dynamic multi-objective problems by using MOEAs. Two procedures to parallelize MOEAs for DMO are proposed in this thesis. The first one, called pdMOEA, is a hybrid approach that can vary from a master-worker paradigm to an island paradigm. A second fully distributed procedure is proposed to improve further the hybrid approach. After describing both procedures, experimental results are provided to assess the performance of both parallel approaches.

- **Chapter 5. Summary Conclusions and Contributions** is the last chapter and it offers a summary of the conclusions brought in this thesis along with the main contributions made and the publications that have been produced. It also includes the future work that opens from this thesis.

In order to help the reader to navigate through this thesis there are some lists that reference and compile certain kinds of information. They are the following ones:

- A **List of Abbreviations**, available at the beginning, where the reader can find the most frequent abbreviations used in this thesis with a description of what they stand for.
- A **List of Figures**, at the end of the thesis, compiles all the Figures that are part of this thesis.
- A **List of Tables** is found after the List of Figures. It provides the reader with an easy way to find any Table that have appeared in the thesis.
- A **List of Algorithms** is given also to help the reader to search for the different algorithms and procedures of this thesis.
- A **Glossary** is provided after the List of Algorithms. It offers descriptions for those new or difficult terms that are scattered along this dissertation.
- Finally, an **Index** gathers together the concepts and terms that have been used in this thesis with an indication of the pages where they have appeared.

The next and first chapter reviews the current state of research in the three fields that play a central role in this thesis. Multi-objective evolutionary algorithms, dynamic problem optimization and parallel approaches for MOEAs are considered. Chapter 1 provides a suitable introduction of the current state of the different fields that will be discussed in the rest of the thesis.

"Knowing is not enough; we must apply. Willing is not enough; we must do."

Johann Wolfgang von Goethe

1 Introduction and Field Review

In this chapter it is given an introduction to the main topics that this thesis addresses. In addition, it offers a detailed review of the current state-of-the-art in those research areas along with the main achievements that have happened in any of these areas.

Therefore, this chapter enables any reader with less experience in any of the areas covered by this thesis to understand this PhD thesis completely.

First of all, in Section 1.1 the concepts and definitions around multi-objective optimization are provided. Secondly, the reader can find in Section 1.2 enough material about evolutionary algorithms, including those aimed at multi-objective optimization problems. The Section begins with a review of alternative metaheuristic optimization methods (Sub-section 1.2.1). Then, attention is paid to the parallelization of multi-objective evolutionary algorithms (Sub-section 1.2.3). In Sub-section 1.2.4 the *No Free Lunch* theorems are introduced in order to have a basis to make comparisons with other metaheuristics. Finally, in Section 1.3 the concepts and ideas behind Dynamic Problem Optimization, the central topic of this thesis, are exposed.

1.1 Multi-objective Optimization Problems

An optimization problem occurs when you have to select the best solution from all feasible ones to a given problem. There are many examples of optimization problems, but two are very popular among researchers:

- *The traveling salesman problem* (TSP), where a salesman has to visit different cities and the total distance that he travels after visiting all the cities and returning to the origin must be minimized by choosing the most efficient or best route to follow. This means that an ordering of the cities must be chosen so that the total distance is minimized.
- *The knapsack problem*, where there is a container, typically a bag, with a certain maximum capacity and some items that could be placed inside that container. Then, the optimization problem is to find which items can be placed in the bag so that they occupy the maximum possible space, or in other words, that leave the least free space.

In both problems, we do not know the *best* solution, or simply *the solution*, to any instance of the problem and what algorithms do is to choose solutions that are better than the solutions found so far until a stopping criterion is reached. Instead of knowing the best solution, we know that there must be at least one solution that is not worse than the rest of solutions. This is so because the solutions are evaluated with a *fitness* or *cost* function whose range lies in $\mathbb{R}$. In the above-mentioned problems, the *fitness* functions are, respectively, the total distance that the salesman has to travel to visit all the cities and the volume of the container that is occupied. Both, distance and volume, are values inside $\mathbb{R}$ and so they both obey the total order binary relation that exists in $\mathbb{R}$.

An optimization problem is defined mathematically in Definition 1.1.

Definition 1.1 *An* optimization problem *is a function* $f(x) : \mathbb{R} \rightarrow \mathbb{R}$ *such that* $\exists\ r \in \mathbb{R} \mid f(r) \leq f(x)\ \forall x \in \mathbb{R}$.

Without loss of generality we assume that $f(x)$ is a minimization optimization problem in the rest of this chapter. Also we have assumed that the optimization problem is defined in the total order binary relation defined over $\mathbb{R}$ but $f(x)$ could also be defined over a partial order binary relation over $\mathbb{R}$. The function $f(x)$ is called the *fitness* or *cost* function for that optimization problem.

Definition 1.2 *The* search space *or* decision space *is the dominion of the variables of the optimization problem function.*

Definition 1.3 *The* objective space *is the set of values that an optimization problem fitness function takes or the values that we are looking to optimize for the problem.*

It is important to point out that in all this PhD thesis the objective and decision spaces equal $\mathbb{R}$ or are some vector space derivated from $\mathbb{R}$. It is possible to have optimization problems defined for sub-spaces of $\mathbb{R}$ such as $\mathbb{Z}$ or $\mathbb{N}$ but we are not going to consider them in this thesis.

Definition 1.4 *The* set of solutions *for $f(x)$ is the set $\{s_1, s_2, \ldots, s_n\}$ such that $f(s_1) = f(r) \wedge f(s_2) = f(r) \wedge \cdots \wedge f(s_n) = f(r)$*[1] *and it is represented by $\mathcal{S}$.*

It has been said that there is at least one solution to these optimization problems and so $|\mathcal{S}| \geq 1$. In many problems there will be only one solution ($|\mathcal{S}| = 1$) but in some others there could be more than one. In that case, any of the solutions from $\mathcal{S}$ will be the best solution and valid for the person interested in solving the problem.

The same reasoning applies to multi-evaluated problems, defined in what follows, whose fitness functions lies in $\mathbb{R}$.

Definition 1.5 *A* multi-evaluated optimization problem *is defined by a function $f(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$ where $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$.*

In other words, these multi-evaluated problems optimize more than one parameter at a time ($\{x_1, x_2, \ldots, x_n\}$) but their fitness function always returns a real value, and so there always be at least one solution that is the best one, mathematically $|\mathcal{S}| \geq 1$.

[1]We remind the reader that $a \wedge b$ is the way of expressing a and b in logic.

In this work we are not interested in this kind of problems because they usually contain a reduction of the multiple variables in the decision space to the sole variable in the objective space. This is because the variables in the decision space do not conflict each other, and so the improvement of one of them does not worsen the rest of decision variables.

The interest of these multi-evaluated optimization problems is that they are the intermediate step among simple optimization problems and another sort of problems: *multi-objective optimization problems*.

The main characteristic of multi-objective optimization problems is that their decision and objective space are vector spaces. Multi-objective optimization problems are very important because most of the problems that engineers face every day are multi-objective. Some examples are:

- The optimization of the design of a cylindrical piece of a mechanical system where the decision variables are the three dimensions of the piece and the objective variables are the size and the strength of it. Here, the size of the piece should minimized in order to minimize the cost of its production, but in doing so, the strength of the piece decreases, while obviously, the designers want to maximize the strength of the pieces (Laumanns and Laumanns, 2005; Benedetti, Farina, and Gobbi, 2006).

- When designing a new tire, many decision variables are taken into account: softness of the material, width of the tire, depth and design of the patterns, etc. At the same time, the tire should maximize different objective variables including duration of the tire under different weather conditions, evacuation of water, stability on different roads, resistance to punctures while minimizing the effect of the weather on its performance (Cho, Jeong, and Yoo, 2002; Koishi and Shida, 2006).

These examples show clearly why multi-objective problems are so difficult to solve: multi-objective optimization problems have to optimize different objective functions that conflict with each other. For example, when minimizing the size of the mechanical piece, the strength of the piece is accordingly worsen. In the case of the tire, the situation is considerably worse for the designers because they have more decision variables to choose from and more objective variables to optimize.

In both cases the designers must choose a compromise or trade-off among all the different objectives that guarantees that the chosen solution will serve well if not excellently for its purpose. In a simplification of the tire problem this could mean that one tire can be produced for less than 20 € while on average it can run for more than 40.000 km.

A formal definition of multi-objective optimization problems is given in Def. 1.6.

Definition 1.6 *A* multi-objective optimization problem (MOP) *is a quadruple* (X, Z, f, rel) *where* X *denotes the* search *or* decision space, Z *represents the* objective space, $f : X \rightarrow Z$ *is a function that assigns to each solution or* decision vector $\mathbf{x} \in X$ *a corresponding* objective vector $\mathbf{z} = f(\mathbf{x}) \in Z$, *and rel is a binary relation over Z that defines a partial order of the objective space.*

Without loss of generality in the rest of this PhD thesis we will assume that $X = \mathbb{R}^n$ and $Z = \mathbb{R}^m$ and that the f function must be minimized for all objectives in Z. Therefore, from definition 1.6, it follows that $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ and $f(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\}$ and our optimization problem becomes the quadruple $(\mathbb{R}^n, \mathbb{R}^m, f, \preceq)$.

The binary relation $\preceq$, presented in what follows, is a natural extension of the total order relation $\leq$.

Definition 1.7 *The binary relation* $\preceq$ *is defined by* $\mathbf{z}^1, \mathbf{z}^2 \in Z : \mathbf{z}^1 \preceq \mathbf{z}^2 \iff \forall i \in \{1, \ldots, m\} : \mathbf{z}^1_i \leq \mathbf{z}^2_i$. *This relation is called* weak Pareto dominance[2].

Definition 1.8 *The binary relation* $\prec$ *is defined by* $\mathbf{z}^1, \mathbf{z}^2 \in Z : \mathbf{z}^1 \prec \mathbf{z}^2 \iff \forall i \in \{1, \ldots, m\} : \mathbf{z}^1_i < \mathbf{z}^2_i$. *This relation is called* strict Pareto dominance.

When $\mathbf{z}^1 \prec \mathbf{z}^2$, it is said that $\mathbf{z}^1$ dominates $\mathbf{z}^2$ or that $\mathbf{z}^2$ is dominated by $\mathbf{z}^1$. The increase in dimensions of the objective space means that there will be probably multiple minimal elements of $f(X)$, where each one will represent a different trade-off between the objectives.

In addition, if $\mathbf{z}^1 \npreceq \mathbf{z}^2$ and $\mathbf{z}^2 \npreceq \mathbf{z}^1$, they are said to be *incomparable* and it is denoted by $\mathbf{z}^1 \parallel \mathbf{z}^2$. This happens when $\mathbf{z}^1$ is strictly better than

[2]Named after the italian economist and sociologist **Vilfredo Pareto**.

$\mathbf{z}^2$ in at least one objective value and $\mathbf{z}^2$ is strictly better than $\mathbf{z}^1$ in at least another different objective value.

Those elements in X that are minimal, i.e. that are not dominated in Z by any other element in X, are called *Pareto optimal* decision vectors, and their images in Z are called *Pareto optimal* objective vectors. All those Pareto optimal vectors define the Pareto optimal front (definition 1.10) and the Pareto optimal set (definition 1.9):

Definition 1.9 *The* Pareto set *is the set of solutions or decision vectors that are minimal:* $\{\mathbf{x}_i \in X \mid \nexists \mathbf{x}_j \in X : f(\mathbf{x}_j) \prec f(\mathbf{x}_i)\}$ *and is denoted by* $\mathbb{S}_p$.

Definition 1.10 *The* Pareto front *is the set of objective vectors of the Pareto set:* $\{f(\mathbf{x}_i) \in Z \mid \nexists \mathbf{x}_j \in X : f(\mathbf{x}_j) \prec f(\mathbf{x}_i)\}$ *and is denoted by* $\mathbb{F}_p$.

One more definition will be useful, the *Pareto set and front approximations* are described in Def. 1.11.

Definition 1.11 Pareto set approximations *are sets of mutually incomparable solutions. Accordingly,* Pareto front approximations *are sets of mutually incomparable objective vectors.*

It is common practice to denote Ψ as the set of all Pareto set approximations over X and Ω as the set of all Pareto front approximations over Z.

In Figure 1.1 the two sets for a two-objective optimization problem are shown. At the left, it is shown the search or decision space, and at the right it is shown the objective space. Four solutions $\{\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, \mathbf{x}^4\}$ are shown with their corresponding location at the objective space after applying them the function $f(\mathbf{x})$.

It can be seen in Figure 1.1 that the location of the vectors at the search space does not hold any correspondence with their location at the objective space.

1.1.1 Further Reading.

There are a few sources that extend the short introduction to multi-objective optimization problems that has been presented in this section.

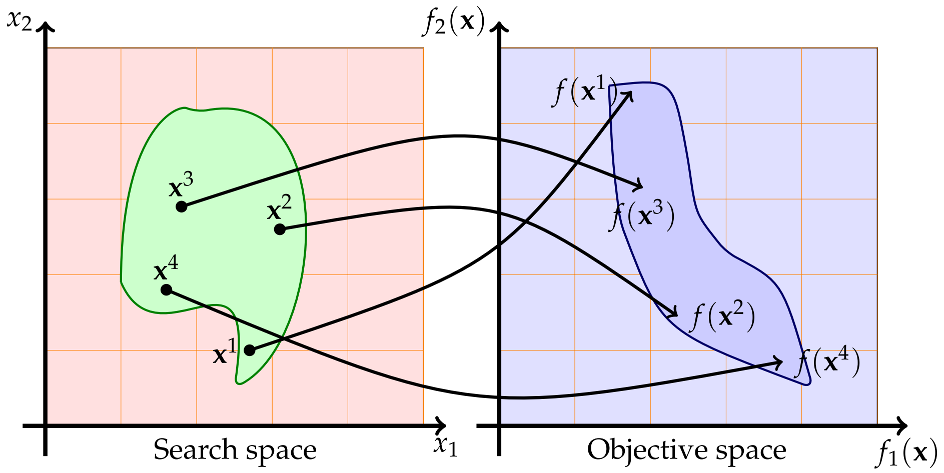

Figure 1.1: Representation of the search and objective spaces in a multi-objective optimization problem.

Good and thorough descriptions are given in Zitzler, Deb, and Thiele (2000); Zitzler, Laumanns, Thiele, Fonseca, and da Fonseca (2002) and Knowles, Thiele, and Zitzler (2006).

Moreover, for a complete and thorough review of the theory behind multi-objective optimization the interested reader is referred to Chapter 6 in Coello, Lamont, and van Veldhuizen (2007).

1.2 Evolutionary Algorithms and Metaheuristics

Evolutionary algorithms are part of a class of algorithms known as metaheuristics. A metaheuristic should be a heuristic that is about heuristics, but here it is used in the sense of a heuristic method used to solve a problem by using other heuristic methods. Sean Luke (Luke, 2009) says that

> *"A metaheuristic is a rather unfortunate term often used to describe a major subfield, indeed the primary subfield, of stochastic optimization."*

or stated in plain words:

> *"[metaheuristics are] algorithms used to find answers to problems when you have very little to help you: you don't know what the optimal solution looks like, you don't know how to go about finding it in a principled way, you have very little heuristic information to go on, and brute-force search is out of the question because the space is too large. But if you're given a candidate solution to your problem, you can test it and assess how good it is. That is, you know a good one when you see it."*

In this way, stochastic optimization is comprised of algorithms used to solve problems by using some degree of randomness. Metaheuristics are also known as *black-box optimization* or *weak methods*. Nonetheless, the most widely accepted term among researchers is metaheuristics.

In this section multi-objective evolutionary algorithms are introduced because they are a central part of this thesis but before delving into details about them, other metaheuristics are reviewed in the next subsection.

1.2.1 Multi-objective Metaheuristics.

Apart from evolutionary algorithms there are other algorithms considered also metaheuristics that could be used to solve Dynamic Multi-objective Optimization (DMO) problems. The most common metaheuristics are:

- Simulated Annealing (Dowsland, 1993), is a stochastic search algorithm that simulates the concept of annealing which is that after a solid temperature is raised to a point where its atoms can freely move, the temperature is gradually lowered in order to force the atoms to crystallize. The crystallization usually becomes when the atoms are in a state of low energy. In this metaheuristic, the energy of the annealing process imitates the fitness of the optimization process, and so a low energy means a low value of fitness which is desired when solving minimization problems.

- Tabu Search (Glover, 1989), where potential solutions are marked as taboo after being found in order to avoid them in the future iterations of the algorithm. In this case, the metaheuristic tries to avoid repetition of local optima. To find new solution local search algorithms are usually employed.

- Scatter Search (Glover and Laguna, 2000) that operates on a set of solutions known as reference set by combining these solutions to create new ones by using rules such as the linear combination of two solutions. Thus, Scatter Search creates new solutions from older ones. In contrast to the big populations used in Genetic Algorithms, in Scatter Search the reference set of solutions tends to be small, usually below 20 solutions. Another difference between Scatter Search and Genetic Algorithms is that in Scatter Search solutions that are to be combined are chosen in a systematic way while in Genetic Algorithms they are chosen randomly.

- Ant Colony Optimization (Dorigo and Stützle, 2004) is inspired by colonies of real ants and how they discover paths to food. They work incrementally to construct a solution by adding solution components to a partial constructed solution.

- Particle Swarm Optimization (Kennedy and Eberhart, 1995). In this case the algorithm try to simulate a flock of birds that are looking for food. In Particle Swarm Optimization each particle (solution) keeps track of its coordinates in the problem space. At each iteration, the algorithm updates the coordinates of the particle by using a acceleration term and the location of the best solutions found so far. Randomness is applied along with the acceleration term used with each particle.

- Differential Evolution (Storn and Price, 1997) that uses a scheme for generating trial parameter vectors without relying in a separate probability function as other metaheuristics do. Differential Evolution optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple formulas of vector-crossover and -mutation, and then keeping whichever candidate solution has the best score or fitness on the optimization problem at hand.

- Artificial Immune Systems (DasGupta, 1998) are intelligent system that are able to learn and retrieve previous knowledge. Artificial

Immune Systems (AIS) are concerned with abstracting the structure and function of the immune system to computational systems, and investigating the application of these systems towards solving computational problems.

1.2.2 Evolutionary Algorithms

Evolutionary algorithms is a family of algorithms that share a common feature, all of them mimic in some way the natural evolution process as it was described by Darwin (Darwin, 1995). This collective name of evolutionary algorithms is comprised of the following kinds of algorithms:

- Genetic algorithms (Mitchell, 1996), where the solutions to a problem are coded as genes. These solutions may then be crossed among themselves or be mutated. Moreover, these algorithms use the concept of selection to choose the best solutions among all the available ones. In genetic algorithms, a set of solutions, called population, is optimized in every generation. If the whole population changes completely in one iteration or *generation*, the Genetic Algorithm is *generational* whilst it is *steady-state* if only a part of the population is changed. Genetic Algorithms are probably the most popular one within the family of evolutionary algorithms.

- Evolution strategies (Beyer and Schwefel, 2002) is mainly differentiated from the genetic algorithms, in that it employs a simple procedure for selecting individuals (Truncation) and usually only uses mutation. In addition, the size of the population is usually smaller than it is for Genetic Algorithms, even it can be comprised of only one parent and one child. These algorithms are termed ES(μ, λ) where μ represents the number of parents that survive and λ is the the number of offspring solutions.

- Genetic programming (GP) (Koza, 1992) is a specialization of Genetic Algorithms where the algorithm evolves computer programs, which are usually represented in tree structures, instead of solutions to an optimization problem. It is used to optimize a population of computer programs according to a fitness function. The programs are intended to solve a computational task. Genetic programming also uses the concepts of mutation and crossover that can be applied to any node of the tree, which implies changes in the nodes that stem from it. In Genetic Programming is widely accepted to represent the programs in Lisp-style tree-structured expressions (Graham, 1993; Seibel, 2004). *Grammatical Evolution* (O'Neill and Ryan, 2003) is a variant where the programs are represented as strings of integers that can be mapped to programs by means of a grammar.

- Evolutionary programming (Fogel, 1999) also evolves a computer program, but contrary to what happens in GP, in evolutionary programming the program structure is fixed and only the actual values inside the nodes are allowed to be evolved, i.e. to change. Its main variation operator is mutation. In Evolutionary Programming, members of the population are viewed as part of a specific species rather than members of the same species therefore each parent produces one child.

The origins of the evolutionary algorithms approach dates back from the 50's when Nils Aall Barricelli first used evolutionary algorithms to carry on simulations inspired in the natural evolution (Barricelli, 1957, 1963a,b). But it was in the 60's and 70's when this family of algorithms saw important breakthroughs with researchers such as Fogel, Rechenberg and Holland. In the 80's, more powerful computers made possible to extend the application of these algorithms and the field attracted major attention from researchers from other areas.

The use of evolutionary algorithms to solve multi-objective optimization problems comes in a natural way because of the use of a population of solutions within the evolutionary algorithms, which is similar to the set of solutions to the multi-objective optimization problem.

In this thesis, we will use only genetic algorithms, however the approach and procedures used could be applied to any evolutionary algorithm from the four afore-mentioned ones.

The first implementation of a multi-objective evolutionary algorithm (MOEA) was made by David Schaffer (Schaffer, 1984) in 1984, when he proposed the *Vector Evaluated Genetic Algorithm* (VEGA) (Schaffer, 1985).

In VEGA, the algorithm optimizes an objective for each solution at a time, avoiding the problem of optimizing all the objectives at the same time.

Almost a decade later, a new approach to solve MOP was initiated with the MOGA algorithm (Fonseca and Fleming, 1993). In it, the concept of dominance rank was introduced in order to optimize all the objectives at the same time. This was a major breakthrough in the field of MOEAs.

Later, other researchers published their proposals for MOEAs. There are two MOEAs that have become the most used algorithms and they still remain today as the reference MOEAs to compare with when developing new ones. Both of them are second improved versions of the original algorithms proposed by their authors.

The first of them is the improved *Non-Dominated Sorting Algorithm II* (NSGA-II) (Deb, Agrawal, Pratap, and Meyarivan, 2000). The second one is the *Strength Pareto Evolutionary Algorithm 2* (SPEA2) (Zitzler, Laumanns, and Thiele, 2002).

Both algorithms share the point that they usually obtain very good sets of solutions while they differ in the way they select the solutions to carry to the next generation. More details about NSGA-II and SPEA2 can be found in Sections 3.1.2 and 3.1.3, respectively.

Another MOEA is the *Single Front Genetic Algorithm* (SFGA) (de Toro, Vidal, Mota, and Ortega, 2002). The interest of this algorithm lies in its simplicity that stems from using only a single front of non-dominated solutions along with a crowding method. The result is a very fast algorithm, that can compete with the afore-mentioned NSGA-II and SPEA2 in the quality of solutions, while it is much faster than they are. This algorithm is fully described in Section 3.1.1. An improved version of SFGA has been developed to deal with DMO problems. This new version is introduced in Chapter 3 of this thesis.

1.2.3 Parallelization of MOEAs.

One important topic related to MOEAs is to parallelize them (Cantu-Paz, 2000). Due to the population that share all the MOEAs, parallelism arises naturally as a way to improve both the performance of the algorithm and the quality of the solutions found. Unfortunately, it is not a straightforward task. Because of the importance of improving the performance and the quality of the solutions, researchers have proposed different approaches to parallelize MOEAs. The goal is that by using parallelism more processes could run the same algorithm at different parts of the search space, or at least, to produce a better set of solutions for the problems. However this is not always achieved, because it is usually quite difficult to guide different instances of a MOEA to search at different parts of the search space.

Two decomposition alternatives are usually implemented in parallel evolutionary algorithms: functional decomposition and data decomposition.

The functional decomposition techniques identify tasks that may be run separately in a concurrent way. The data decomposition techniques divide the sequential algorithm in different tasks that are run on different data (i.e. the individuals of the population). Moreover, hybrids methods are also possible.

In an evolutionary algorithm, the evaluation of the objective function and the application of operators to the individuals of the population can be independently done for each individual. This allows data parallelization without modifying the convergence behavior of the sequential algorithm. The fitness evaluation for each individual in the population is usually the part with the highest computational cost. This is mainly true in non-trivial optimization problems, with big sized populations and/or individuals codified with complex data structures that require big computation times.

As a consequence, the most usual parallelization scheme is to evaluate concurrently the individuals, usually with a master-worker implementation in which every worker process evaluates a different and unique group of individuals, returning the fitness values to the master process which continues with the rest of the algorithm steps.

The parallelism of MOEAs and, more concretely, parallelism applied to DMO, is one of the main topics of this thesis.

Parallel MOEAs are usually classified into three different kinds of paradigm according to the structure adopted by the processes (Tomassini, 1999; Alba, 1999; Cantu-Paz, 2000):

- *Master-worker* or *master-slave* paradigm, when one process is the *master* because it controls the execution of the other processes which are the *workers*. The master process uses a population that it is the sum of the populations of the worker processes.

- *Island* paradigm, where there is not a master process and all the workers work independently. Again, the workers can search in different areas or share the same area.

- *Cellular* or *diffusion* paradigm, where only one individual of the population is allocated to each process. This is a very fine-grained parallel approach. Grid structures are designed to allow the processes to exchange their solution with their neighbours.

Several authors include a fourth kind which is a combination or hybrid approach of the other three (Coello et al., 2007).

Returning back to the master-worker model, if the individuals are distributed in a balanced way there could be linear speedups, but unless the evaluation of the solutions require a high computation time, the costs associated with the distribution of the data structures between the processors and the communication of the results may considerably decrease the efficiency of this kind of parallel procedures.

The selection of individuals and the diversity maintenance operations require comparisons that imply the whole population or a big part of it. This means that data parallelization at this level, specially in the case where there is not any mechanism to share information among the processes about the fitness of the individuals, modifies the behaviour of the algorithm with regard to the sequential version.

Usually, it is difficult to predict the behaviour of this kind of parallelization and must be evaluated for each particular implementation. The initial population is divided into sub-populations associated to different search spaces which are evolved separately.

Sometimes, individuals can be exchanged between the sub-populations (migration). This kind of parallelization could improve the diversity of the population during the algorithm convergence and lead to algorithms with better performance than the sequential versions.

So, together with the advantages due to the bigger availability of memory and CPU, the evidences of bigger efficiency and diversity in the population justify the use of parallelism in the field of evolutionary algorithms.

After the evaluation of the objective functions, the algorithms with Pareto front-based selection usually calculate dominance and the corresponding distances as part of the mechanism for keeping diversity. This mechanism is implemented in each case, as a previous step to assign fitness values to each individual and to select the parents. The parallelization of these tasks is not easy. For example, problems appear in algorithms that usually work with small populations, PAES (Knowles and Corne, 1999), in algorithms where the calculation of distances must be done sequentially after the determination of dominance relations, PSFGA (de Toro, Ortega, Ros, Mota, Paechter, and Martín, 2004), or in those algorithms where the calculation of dominance relations and distances, and the selection take place at the same time, NPGA (Horn and Nafpliotis, 1993).

Some approaches to parallelize MOEAs have been proposed in the past but they are all for multi-objective stationary problems (de Toro et al., 2002; Alba, 2005).

Chapter 4 of this thesis is devoted to parallelism aimed at Dynamic Multi-objective Optimization by using MOEAs in the processes.

1.2.4 The No Free Lunch Theorems.

One important issue about evolutionary algorithms is that they are metaheuristic methods to search for a solution to a given problem. This poses the question about how reliable they can be and how they compare to other algorithms.

To answer this question two researchers provided the *No Free Lunch* (NFL) theorems (Wolpert and Macready, 1995, 1997; Köppen, 2004).

Broadly speaking, the NFL theorems states that given two metaheuristics A and B, if A performs better than B on some problems, then B performs better than A in other problems in such a way that both perform average over the set of all problems or put in Wolpert's and Macready's plain language *"any two algorithms are equivalent when their performance is averaged across all possible problems."*(Wolpert and Macready, 2005).

David Corne and Joshua Knowles have proposed an extension of the NFL theorems for multi-objective optimization problems in Corne and Knowles (2003). The outcome of this extension for DMO is what we already know, that one MOEA can outperform some other MOEAs for an instance of a problem, while in the next run the first MOEA is outperformed by any of the losers from the first round.

This is a reminder that any result about performance shown in this thesis, even after carefully designing the experiments followed by thoughtful statistical analysis, will be just an indicator about an algorithm performing better than others with respect to a given problem.

After seeing all those different metaheuristics in subsection 1.2.1 a question arises naturally from the NFL theorems and their extensions. We have said that MOEAs are very good to solve DMO problems, but *is there any other metaheuristic that performs on average better than MOEAs do?*. The short answer is that probably there are. But if we draw from the experience of the last two decades of development of MOEAs, we can state that MOEAs are one of the best metaheuristic for solving multi-objective problems up-to-date.

In this thesis, MOEAs are used also for dynamic problems and it is shown that their behaviour is also satisfactory.

1.2.5 Further Reading.

There are two reference books in the field of multi-objective evolutionary optimization: Coello et al. (2007) and Deb (2001). Moreover, Coello also maintains a website where he compiles literature related to multi-objective optimization with evolutionary algorithms (Coello, 2008).

More information about parallelization of MOEAs can be found in Cantu-Paz (2000); van Veldhuizen, Zydallis, and Lamont (2003). In Luna, Nebro, and Alba (2006), a complete review of parallel evolutionary algorithms can be found. Additional topics can be consulted in Nedjah, de Macedo Mourelle, and Alba (2006).

Parallel implementations of other kind of metaheuristics are found in Alba (2005); Talbi (2006) and Talbi (2009).

Specific proposed procedures are PSFGA (de Toro et al., 2002, 2004), MOSATS (Baños, Gil, Paechter, and Ortega, 2006, 2007; Baños, Gil, Reca, and Ortega, 2010) and pdMOEA, the procedure proposed in this thesis (Cámara, Ortega, and de Toro, 2007a, 2008b, 2010).

Researchers can make use of the ParadisEO library (Cahon and Talbi, 2004) to develop programs with off-the-shelf implementations of parallel and sequential metaheuristics. This is an extension of the Evolving

Objects (EO) library (Keijzer, Merelo, Romero, and Schoenauer, 2002).

A current update on the NFL theorems and their implications is given in Rowe, Vose, and Wright (2009).

The interested reader in other metaheuristics presented in Sub-section 1.2.1 is referred to a freely available book (Luke, 2009) and again to Talbi (2009). Additional information on certain topics can be found in Glover and Kochenberger (2003).

1.3 Dynamic Optimization Problems

In this section dynamic problems are introduced together with some notation regarding to them.

Roughly speaking, *dynamic optimization problems* (Cámara et al., 2007a; Farina, Deb, and Amato, 2004; Jin and Branke, 2005) are those problems where the restrictions or the objective functions of the problem change with time, and so the solutions obtained at time t could not be the correct solutions for time $t + \delta$.

Definition 1.12 *A* dynamic multi-objective optimization problem *(DMO) is defined as the quintuple (X, Z, T, f, rel) where X denotes the* search *or* decision space*, Z represents the* objective space*, T is the time domain, $f : T \cup X \longrightarrow Z$ is a function that assigns to each solution or* decision vector $\mathbf{x} \in X$ *and each value of time $t \in T$ a corresponding* objective vector $\mathbf{z} = f(\mathbf{x}) \in Z$*, and rel is a binary relation over Z that defines a partial order of the objective space.*

Therefore, the main difference with respect to MOP (definition 1.6) is that time has been added to the optimization process. Thus, we have to find a decision vector $\mathbf{x}^*(t) = \{x_1^*(t), x_2^*(t), \ldots, x_n^*(t)\}$ that optimizes the function vector: $f(\mathbf{x}, t) = \{f_i(\mathbf{x}, t) \ : 1 \leq i \leq m\}$ where t represents the time or the dynamic nature of the problem.

Accordingly, definitions 1.9 and 1.10 are modified to include the notion of time.

Definition 1.13 *The* Pareto set *at time t of a DMO is the set of solutions or decision vectors that are minimal at time t: $\{\mathbf{x}_i \in X \mid \nexists \mathbf{x}_j \in X : f(\mathbf{x}_j, t) \prec f(\mathbf{x}_i, t)\}$ and is denoted by $\mathbb{S}_p(t)$.*

Definition 1.14 *The* Pareto front *at time t of a DMO is the set of objective vectors of the Pareto set at time t: $\{f(\mathbf{x}_i) \in Z \mid \nexists \mathbf{x}_j \in X : f(\mathbf{x}_j, t) \prec f(\mathbf{x}_i, t)\}$ and is denoted by $\mathbb{F}_p(t)$.*

In other words, for a DMO problem, $\mathbb{S}_p(t)$ and $\mathbb{F}_p(t)$ are the sets of Pareto optimal solutions at time t, respectively, in the decision and objective spaces. A classification of DMO problems (Farina et al., 2004) depending on whether the sets $\mathbb{S}_p(t)$ and $\mathbb{F}_p(t)$ change with time or not is given in Table 1.1.

Table 1.1: Types of dynamic problems depending on time

	Pareto set $\mathbb{S}_p$	
Pareto front $\mathbb{F}_p$	Changes with t	No changes with t
No changes with t	Type I	Type IV
Changes with t	Type II	Type III

It can be seen from Table 1.1 that Type IV problems are those where there is not any change along the time. They are known as plain multi-objective optimization problems or stationary problems.

In this thesis we are interested in solving the problems that fall under the Type I, II and III categories. When designing and testing algorithms there are not differences whether the problem is of Type I, II or III, because all of them imply a change of the solutions and the Pareto front whenever the underlying problem says to do so. Because of this, in this thesis, when referring to DMO problems, it means any Type I, II or III problem.

1.3.1 Classification of Problems.

We have already seen the differences between *stationary* or *static* optimization and *non-stationary* or dynamic optimization. Along with that characteristic, we have seen in Section 1.1, that optimization problems can also be divided as being single-objective or multi-objective.

Table 1.2 shows another classification of the problems that unifies the two features seen earlier. In this way, problems are classified depending on whether they are single or multi-objective and stationary or dynamic.

Each combination gives a type of problem identified by a letter. In brackets it is shown the usual abbreviation that is used to refer to this kind of problems.

Table 1.2: Classification of a problem according to its nature

	Single objective	Multi-objective
Stationary	Type A (OP)	Type B (MOP)
Dynamic	Type C (DOP)	Type D (DMO)

1.3.2 Evolutionary Algorithms and DMO.

The use of evolutionary algorithms to solve dynamic or non-stationary problems was pioneered in Goldberg and Smith (1987) twenty years ago. Since then, this topic slowly drew some attention and it was in the second half of the nineties when multiple works about this subject saw the light like those in Vavak, Fogarty, and Jukes (1996); Mori, Kita, and Nishikawa (1998); Grefenstette (1999) and Trojanowski and Michalewicz (1999), among many others.

Due to this, the field has grown with many ramifications, depending on the particular approach taken by the researchers. In what follows, it is presented a summary of the main approaches to dynamic problems published in the last years.

Jürgen Branke, a prolific researcher in this topic, has produced some important publications in the subject of dynamic optimization. The most important ones are:

- He maintains an online bibliography (Branke, 2008) on the subject.

- In Branke and Schmeck (2003), the authors give a fair introduction on how to design evolutionary algorithms to deal with dynamic problems.

- Branke has proposed approaches to dynamic problems based in memory enhancing (Branke, 1999b).

- Moreover, he has discussed the use of multiple populations (Branke, Kauler, Schmidt, and Schmeck, 2000) in order to improve the solutions found for dynamic problems.
- Branke and Mattfeld (Branke and Mattfeld, 2005) introduce the concept of flexibility as a desirable feature for dynamic scheduling and the way to deal with it by using an evolutionary algorithm.
- An analysis of some theoretical issues concerning dynamic problems is given in Branke, Salihoğlu, and Şima Uyar (2005).
- Finally, Yin and Branke have given an overview of uncertainty in dynamic problems using evolutionary algorithms (Jin and Branke, 2005).

Peter Bosman (Bosman, 2005; Bosman and Poutré, 2007) has studied the presence of the time-linkage problem in dynamic optimization both in theoretical and practical ways.

The time-linkage problem appears when decisions taken at the present affect the reachable solutions in the future. The cause of this situation is that time-linkage may deceive an algorithm making it to find only suboptimal solution trajectories. Bosman has addressed this problem by trying to learn from the past in order to predict the future.

Following a similar reasoning to Bossman in Rossi, Abderrahim, and Díaz (2008) is proposed the use of Kalman filters in order to bias the search for tracking a changing optimum in dynamical optimization problems making use of the information provided by a prediction mechanism. Again, the prediction mechanism is based on the assumption that in real world applications changes are not random and can be learned.

In Bui, Nguyen, Branke, and Abbass (2007), the authors propose to create a multi-objective optimization problem from a single objective dynamic problem and solve it with a state-of-the-art MOEA (more specifically, NSGA-II (Deb et al., 2000) was used) with the aim of maintaining greater population diversity and adaptability. This is one of the first encounters of multi-objectivity and dynamic problems, although in this case, the problem remains single-objective and the multi-objectivity is introduced as a mean to solve the single-objective dynamic problem.

Another prolific author on dynamic optimization problems is Yang, who has multiple publications on this issue. The most relevant publications are:

- Tinós and Yang (2007) where random immigrants are introduced into a genetic algorithm to tackle dynamic problems;
- Yang (2008) where the immigrant approach is improved by means elitism and memory techniques; and
- Wang, Wang, and Yang (2009), where dynamic problems are solved with a memetic algorithm which is composed of an adaptative hill climbing algorithm and an evolutionary algorithm.

Despite a large quantity of literature can be found on the issue of dynamic problem optimization as it has been previously shown, most of it is only addressed to problems with only one objective functions. However, the optimization problems in real world applications rarely depend on only one objective.

In addition, stationary multi-objective optimization is a topic which has reached an astonishing level of maturity and development through complex theoretical and practical breakthroughs. Similar findings and supporting theories, such as available performance measures, test cases and other theoretical concepts, are missing in the current literature on DMO. In this thesis, the gap is bridged partially by bringing in some analysis, theory and performance measures for DMO. The work done in this thesis has contributed at the beginnings of this field with some publications (Cámara et al., 2007a; Cámara, Ortega, and de Toro, 2007b).

Hatzakis's paper (Hatzakis and Wallace, 2006) is one of the few papers explicitly dedicated to DMO. It shows the use of a prediction mechanism to improve the quality of the solutions found.

Another paper about the subject is Deb, Udaya Bhaskara Rao N, and Karthik (2007) where the authors make slight modifications to the state-of-the-art NSGA-II in order to use it to solve DMO problems.

1.3.3 Further Reading.

There are some sources to look for more information on the subject. Firstly, however a somewhat outdated publication, Branke's PhD thesis (Branke, 2001) is still a good and comprehensive source for dynamic single-objective optimization problems.

In Blackwell and Branke (2006), the authors explain how to use particle swarm optimization to solve dynamic problems. Yang, Ong, and Jin (2007) is a compilation book that offers in a single volume a snapshot of the state-of-the-art research in the subject.

Additional sources aimed only at DMO problems are Farina et al. (2004); Cámara et al. (2007a); Cámara, Ortega, and de Toro (2008c) and Cámara et al. (2010).

Grammatical Evolution was introduced in Sub-section 1.2.2 as a variant of Genetic Programming. In Dempsey, O'Neill, and Brabazon (2009), the reader can find how to tackle dynamic problems with Grammatical Evolution techniques.

Finally, a book (Goh and Tan, 2009) fully dedicated to a slightly different subject, multi-objective optimization in uncertain environments, has been recently published. Even though, the book is recommended for any interested reader on the subject of dynamic optimization, and certainly it may become an important book within the DMO field in the years to come.

1.4 Summary

In this chapter, the current state of research in the three most important topics upon which this PhD thesis is built have been reviewed. In addition, references for further reading have been given for any reader wanting to broaden his/her knowledge on the subject.

It is interesting to note the time frameworks in which have happened the evolution of the main topics of this thesis. They are summarized in what follows:

- Multi-objective evolutionary algorithms. This is the subject with a longer research trajectory. Indeed, VEGA, the first MOEA, was published in 1984 by Schaffer (Schaffer, 1984). Moreover, the most widely used MOEAs NSGA-II (Deb et al., 2000) and SPEA2 (Zitzler, Laumanns, and Thiele, 2001) were first published, respectively, in

2000 and in 2001. In addition, SFGA (de Toro et al., 2002), the basis to the work published in this thesis, was published in 2002.

- The field of dynamic problems has also been studied since the late eighties, starting with single-objective dynamic problems. Nevertheless, it has been only in the last five years that interest has emerged into the study of dynamic optimization problems that are not only dynamic but also multi-objective. Thus, this field is still young and, consequently, it offers many open questions to explore.

- The parallelization of evolutionary algorithms has been attracting the interest of researchers since the end of the nineties (Alba, 1999; Tomassini, 1999; van Veldhuizen, 1999). In addition, it is a subject that has seen many publications in the ten-years gap until nowadays (de Toro et al., 2002; van Veldhuizen et al., 2003; Cahon and Talbi, 2004; Alba, 2005; Luna et al., 2006; Talbi, 2006; Alba, Dorronsoro, Luna, Nebro, Bouvry, and Hogie, 2007; Cámara et al., 2007a; Luna, Nebro, Alba, and Durillo, 2008; Bui, Abbass, and Essam, 2009).

As it can be seen the main topics of this thesis have different maturity levels ranging from just a few years as it is the case for DMO to more than twenty years as it happens with the field of MOEAs. This has been taken into account when working in this thesis in order to provide a framework that unifies the achievements from any of the three fields and that will allow further development both for the combined field of parallel processing for solving DMO problems by using MOEAs along with development of any of the independent fields.

It has been seen that the main issue of this PhD thesis is to develop a complete and accurate algorithm to solve dynamic multi-objective optimization problems.

Nevertheless, in order to assess if that goal is accomplished, performance measures must be used to compare the new designed algorithms with older ones and to see, and probably to quantify, the differences among the solutions obtained by the various algorithms. That raises the idea that adequate performance measures must be provided if new algorithms are sought. Because of this, the next chapter is entirely dedicated to performance measures aimed at algorithms for dynamic multi-objective optimization.

"He who seeks for methods without having a definite problem in mind seeks in the most part in vain."
David Hilbert

2 Performance Evaluation

IT is a natural condition in we, humans, to look around our environment and to compare the people that surrounds us by attributing them tags or oversimplifying their achievements to something that can be expressed with only a few words: *"He is better paid than I am paid/Her new car is more awesome than mine"*.

Because we, researchers, are also humans, we also have that same need of comparing those things in which we are working. Fortunately, we do this in the sake of further development of science. Indeed, we do not only content ourselves with just a simple comparison between algorithms to know which one of them is the fastest one, but also we demand a precise description of how much faster is an algorithm in comparison to another, or how better the solutions found by our cutting edge new algorithm are in comparison to all other algorithms published until date.

It has passed very much time since the inceptions of mathematics by those Greeks who started to study the relations between geometrical objects such as points, lines and polygons, and devised a system to compare them. Nowadays, the measurement theory is a fully developed mature subfield of mathematics and even their basic axioms demand a high-level of knowledge.

Seen from that point of view, current-day researchers on metaheuristics and bio-inspired algorithms are yet in a position which seems closer to the rudimentary achievements of those Greek pioneers than to the insight available in current mathematics.

One of the most respected books on MOEA includes a chapter wholly dedicated to theory for this kind of algorithms (Coello et al., 2007, Chap. 6). In it, the authors state that even though there is already some theory developed for MOEAs, more advances should be done in order to allow the researchers to better exploit these algorithms. When working on MOEAs for dynamic problems, researchers have even less available theory. Trying to solve partially this lack, this chapter contributes to the development of some theory for MOEAs used with DMO problems.

A very important topic when improving or developing new algorithms is to have a suite of performance measures that can be used to assess whether an algorithm behaves better than another. Despite of the fact that in static multi-objective optimization is not feasible to have a definitive set of such performance measures (Zitzler et al., 2002), some quality measures serving the same purpose have been created.

However, in addition to performance indicators like those for stationary multi-objective algorithms, dynamic problem optimizers need another type of performance measures. These other measures must allow algorithm designers to pick one algorithm from a set of them when deploying an evolutionary computation system to solve a real world problem by pointing out which algorithm of the available ones best suits the current needs, and if it would be able to cope on time with the problem at hand. This preliminary study should take place before the commercial exploitation of the system begins. Because of that, it would allow the designer to use *offline* measures instead of *online* or *on-the-fly* measures.

First, a definition of what performance measures are it is given in Section 2.1. Then, in Subsection 2.1.1, a compilation of the different classifications that can be applied to performance measures is offered, which is followed by a call to researchers to use a common and correct terminology in the literature. After that, in Section 2.2, the author justifies his proposal for new performance measures. Section 2.3 is dedicated to a thorough review of the state of the current literature with regard to performance measures for single and multi-objective dynamic optimization problems. Section 2.4 contains the main contribution made in this chapter: an adaptation of performance measures for single objective dynamic problems to multi-objective ones, where special attention has been paid to those multi-objective dynamic optimization problems with unknown Pareto fronts. Finally, in Section 2.5 the standard test cases for DMO are presented along with modifications for them in order to solve some problems that they had.

2.1 Performance Measures

Measures, as defined mathematically, are systematic ways to assign a single value to a set of N-dimensional vectors. In measure theory, this is interpreted to be the size of the set, and it can be the length of a segment, the area of a surface or the integral of a curve. But in order to assess the performance of optimization algorithms, our interest is that this value can give valuable insight into which algorithm has performed better and by how much.

Definition 2.1 *A* measure *is a function μ defined over a domain Σ on $\mathbb{R}$, that assigns a real value to a set $\mathbf{X} \subset \Sigma$.*

Definition 2.2 *A* performance measure *is a* measure *(Def. 2.1) that takes as its input a set of solutions to a problem obtained by using one or more algorithms and gives as output a real-valued quantification of the quality of those solutions to the solved problem.*

2.1.1 Classification of the Performance Measures.

Since researchers are interested in different aspects of dynamic multi-objective optimization algorithms, at least one performance measure should be available for any of those features that have to be studied, i.e. characteristics of approximation sets, ... Hence, measures should be specifically designed to address the suitability of the algorithm by focusing on only one of the different questions. From that it follows that a first classification of the performance measures can be done with respect to the feature on which they are focused or to the main question that they are trying to answer. Such classification is:

- *Diversity* in the distribution of the solutions in the Pareto front. Thus, this kind of measure gives researchers an idea of how good the solutions are distributed over the search space. A well distributed set of solutions is that one where there are solutions representing the whole Pareto front not only some parts of it.
- *Accuracy* or *Closeness* of the solutions found to real Pareto fronts. On the contrary to *diversity*, *accuracy* tells the researcher how close the found solutions are to the real Pareto front, while ignoring how much of the Pareto front is covered. However, a good value for *accuracy* will imply a good diversity of the approximation set to the real Pareto front in most cases.
- *Stability* of the algorithm. This measure indicates to the researcher how well an algorithm has recovered after a change has happened in the conditions.
- *Speed* is the time spent by the algorithm to solve a problem for a given stopping criterion. In spite of its apparent simplicity, we should not forget it in any classification of performance measures.
- *Reaction time* of the algorithm refers to the number of time units or iterations needed by the algorithm to recover the solutions with a certain level of quality after a change in the problem has taken place.
- *Throughput* or *Yield* of the solutions produced by the algorithm. This is a new measure category that is being proposed in this work for the first time in the literature. *Throughput* indicates the number of solutions that the algorithm produces for each unit of time. Thus, it addresses a very important concern that arises when dealing with dynamic problems, namely, to obtain a sufficiently large number of available solutions per unit of time.

As it can be seen, any of the afore-mentioned *fundamental* measures is focused only on one characteristic of the algorithm and the solutions that it produces. In addition, some measures can be defined in terms of the rest of the measures. For example, *throughput* is defined by the number of solutions found and by the time taken to find them.

Another way to classify the performance measures is done with regard to the moment when they are calculated. Thus, those performance measures that can be calculated during the execution of the algorithm are called *online* measures, whereas those ones that can be only calculated when the algorithm has finished are called *offline* measures. Thus, if a measure as the *accuracy* is calculated after the completion of every iteration of the algorithm, it is *online*, but if it is calculated when the algorithm has completed all the iterations, it is *offline*. However, there are subsets of measures that can be only *offline* because they always require information from the following steps of the optimization algorithm. Therefore, those measures can be only computed after the completion of the whole run of the algorithm.

A third way to classify performance measures is done according to the need of additional further knowledge not provided exclusively by the results from the execution of the algorithm. Thus, the measures are either *dependent*, if they need further information to be calculated, or *independent*, if they can be calculated directly from the results gathered by the algorithm.

This last classification resembles another one found in Weicker (2002). This was done by classifying the performance measures by the knowledge needed about the current optima and it is as follows:

- *Full-knowledge* whether information on the current optima must be available at any time during the running of the algorithm,
- *Partial-knowledge* whether the global optima (best fitness value) has to be known only at certain time instants during the running of the algorithm, and
- *Zero-knowledge* whether it is not necessary any knowledge of the best fitness values from other time instants or executions of the algorithm.

To conclude, Figure 2.1 shows a tree-structure which summarizes the possible classifications for *performance measures* that have been proposed up-to-date.

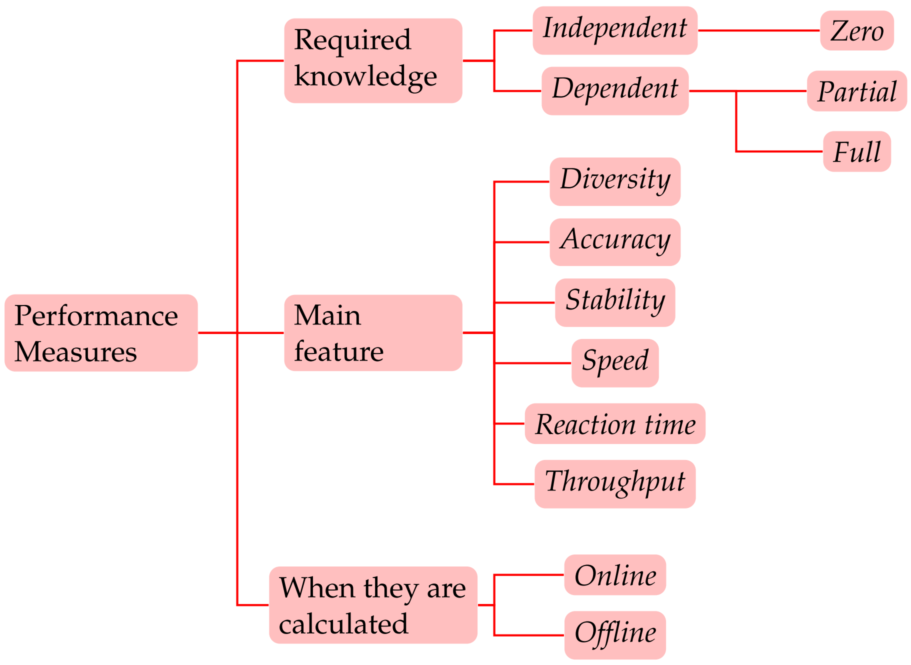

Figure 2.1: Proposed classification of the *performance measures*.

2.1.2 A Note on the Terminology.

There is not a general consensus among researchers on how to refer to the *performance measures*. Some prefer to call them just *(performance) measures* while others prefer to call them *quality indicators* or *quality measures*. In Zitzler, Thiele, Laumanns, Fonseca, and Grunert da Fonseca (2003), it is proposed the term *quality measure* because quality measures are indeed able not only to indicate whether an approximation set is better than another one but also to quantify that difference.

They can be indeed termed as *performance*, or *quality*, *measures* or just *measures*. The important point is that all researchers agree in what term to use. Moreover, independently of the term used, these measures should allow researchers to evaluate just a basic or fundamental part or nature of an optimization problem.

Nevertheless, it is important to remark that the term *metric* is incorrect and should be avoided in the literature because these measures do not represent a *metric* in its mathematical meaning (Sierpinski, 2000).

Definition 2.3 *A* metric, *also called a* distance function, *on a set X is a function $d\colon X \times X \to \mathbb{R}$, where $\forall x, y, z \in X$, d must satisfy:*

1. $d(x, y) \geq 0$
2. $d(x, y) = 0$ iff $x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, z) \leq d(x, y) + d(y, z)$.

The reason that researchers are confused with what a *metric* is stems from the fact that they use it with approximation sets. For example, given two approximation sets A_{Alg_1} and B_{Alg_2}, obtained with algorithms *Alg1* and *Alg2* respectively, it is customary to use a so-called *metric* z such that $z(A_{Alg_1}, B_{Alg_2})$ gives a real value indicating if A_{Alg_1} is better, worse or equal than B_{Alg_2}. As it has seen in Def. 2.3 this is incorrect for two reasons:

1. a *metric* only works with elements within a set, and
2. a *metric* cannot give negative values.

This use of z could be valid if the following circumstance would always hold, namely that researchers would reach an agreement that using an authentic metric on two input sets would have the meaning of unequivocally selecting one element of each set and applying the metric on these two elements. This unequivocal selection could produce the maximum or minimum element within each set, but it relies on a partial order inside the set. It is obvious that this agreement can not be reached for multi-objective problems where an approximation set is meant to contain elements which are not ordered between themselves.

What researchers are really meaning when using $z(A_{Alg_1}, B_{Alg_2})$ is to omit one step from the calculation. This omission gives origin to the incorrect use of the word metric. It can be seen in the following:

$$z(A_{Alg_1}, B_{Alg_2}) \text{ should be}$$
$$z(pm(A_{Alg_1}), pm(B_{Alg_2}))$$
where pm is a performance measure (Def. 2.1).

Researchers obtain the desired result using performance measures, but this use of performance measures and not metrics should be elucidated in the literature. In sum, these measures can be referred to in any of the different proposed forms except for *metric* because it is misleading and wrong. In this thesis, the preferred form is *performance measure*.

2.2 The Need for New Measures

The main contribution of this chapter is the introduction of new performance measures for algorithms aimed at dynamic and multi-objective problems. But before seeing them in detail, it should be justified why they are necessary.

In Table 2.1 there is a summarized reproduction of Table 1.2, where the optimization problems are divided into four different groups.

Table 2.1: Classification of the problems according to their nature

	Single objective	Multi-objective
Stationary	I	II
Dynamic	III	IV

With respect to stationary multi-objective optimization problems, group II, there is a very comprehensive literature that can be looked up (Fonseca and Fleming, 1996; Knowles and Corne, 2002; Knowles et al., 2006; van Veldhuizen and Lamont, 2000; Zitzler et al., 2003). Albeit, there is not such an equivalent development of corresponding measures for problems of the group IV, namely dynamic multi-objective optimization problems.

Indeed, the measures that are lacking are only those which can be classified as related to the dynamic nature of the problem, since the multi-objective part makes use of the already developed performance measures for problems of group II.

Taking into account the dual nature of the dynamic and multi-objective problems, the performance measures classes given in Sect. 2.1.1 can be further divided into three groups, depending on which part of the algorithm they are more related to. This division is made according to the following criteria:

- *Dynamic nature of the algorithm. Stability*, for example, is focused only on the dynamic side of a problem.
- *Multi-objective part of the algorithm.* On the contrary to *stability*, *diversity* is only interested in the multi-objective feature of a problem.
- *Both at the same time.* For example, since algorithms should be fast, *speed* is a performance measure that is important to any of the two faces of the problem.

In Table 2.2 the classes of performance measures seen earlier are assigned to their corresponding category.

Table 2.2: Performance measures according to their fundamental nature

	Multi-objective	Dynamic	Both
Diversity	✓		
Accuracy			✓
Speed			✓
Stability		✓	
Reaction Time		✓	
Throughput		✓	

It has been noted previously in this chapter that the literature lacks an in-depth treatment of measures specifically aimed at group VI problems and their dynamic essence. It can be drawn from Table 2.2 that the measures *stability*, *reaction time* and *throughput*, are important only when solving dynamic problems. Moreover, the measure *accuracy* is also of enormous importance not only when dealing with *stationary* multi-objective problems, but also with dynamic ones. Hence, the main contribution of this chapter is to provide new measures that will cover those needs as it has been outlined in Table 2.2, namely measures for *accuracy*, *stability*, *reaction time* and *throughput* for multi-objective dynamic optimization problems.

It is worth noting that these new measures just supplement those which were available earlier. In fact, these new measures are new tools added to the researcher's swiss army knife, in such a way that his toolkit is empowered, enabling him to study properly those algorithms used with dynamic multi-objective optimization problems.

2.3 Review of the Literature

Despite the large quantity of literature that can be found on the issue of dynamic problem optimization, most of it only addresses problems with only one objective functions (group III problems). However, the optimization problems in real world applications rarely depend on only one objective. In addition, stationary multi-objective optimization (group II problems) is a topic which has reached a certain level of maturity and development through complex theoretical and practical breakthroughs (Coello et al., 2007; Deb, 2001; Knowles and Corne, 2002; van Veldhuizen, 1999; Zitzler et al., 2000; Auger, Bader, Brockhoff, and Zitzler, 2009). Similar findings and supporting theories are missing in the current literature on DMO. This lack of literature is especially noteworthy for DMO performance measures, even though they are the founding bricks on which the rest of the research has to be settled.

The author of this thesis has tried to respect to the maximum the terminology used by every author. Because of this the same concept can be named differently or the same letter can be used to name very different concepts. For example, Morrison (2003) uses G to indicate the number of generations completed by the algorithm, while it is customary in the multi-objective literature to use G to denote the *generational distance* (van Veldhuizen, 1999).

2.3.1 Measures for Single Objective Problems.

As it has been said, measures for dynamic problems, independently of being addressed to single objective or multi-objective problems, have received little attention in literature. The few proposals in this topic are reviewed in the rest of this section, beginning with an account of measures for single objective dynamic problems (group I) which is followed by the measures that have been proposed for multi-objective dynamic problems (group IV) so far.

Morrison's Measures.

Ronald Morrison offers a review (Morrison, 2003) where he suggests which measures taken from stationary algorithms should not be used for dynamic problems. Furthermore, he points out which of those *suitable* measures have been used for single objective dynamic problems so far. These measures can be summarized as follows:

- the difference between the current solution and the best solution just in the previous time step of the problem (Trojanowski and Michalewicz, 1999),
- an *offline* performance measure, where the best-so-far value is reset at each fitness landscape change (Branke, 2001),
- the average Euclidean distance to the optimum at each generation (Weicker and Weicker, 1999),
- best-of-generation[1] averages for many runs of the same problem (Bäck, 1998; Gaspar, 1999; Grefenstette, 1999), and
- the best-of-generation minus the worst one within a small window of recent generations, compared to the best within the window minus the worst within the window (Weicker, 2002).

Morrison ranks the fourth, *"the best-of-generation averages, as the most reported measure but it does not provide a convenient method for comparing*

[1]Morrison uses this expression to refer to the averaged value of the best solutions found in each generation.

performance across the full range of landscape dynamics, nor measuring the statistical significance of the results".

He also states that *"a good performance measurement method for EAs in dynamic environments should, at a minimum, have: (1) intuitive meaning; (2) straightforward methods for statistical significance testing of comparative results; and (3) a measurement over a sufficiently large exposure to the landscape dynamics so as to reduce the potential of misleading results caused by examination of only small portions of the possible problem dynamics"*.

Finally, he introduces two new measures. The first one is the *Total Mean Fitness* F_T, which is the average best-of-generation values over an infinite number of generations, further averaged over multiple runs, or:

$$F_T = \frac{\sum_{m=1}^{M} \left(\frac{\sum_{g=1}^{G} (F_{BG})}{G} \right)}{M} \qquad (2.1)$$
$$\text{with } F_T = \text{constant when } G \rightarrow \text{inf}$$

where:

- F_T is the total average fitness of the algorithm over its exposure to all the possible landscape dynamics,
- F_{BG} is the best-of-generation fitness,
- M is the number of runs of the algorithm, and
- G is the number of generations.

Because G tends to infinity, Morrison says that F_T will not be affected by variation in the best–of-generation fitness values at any given generation of the algorithm.

Additionally, he says that large experiments are not required in order to use this performance measure because the F_T value for evolutionary algorithms approaches a constant after a small representation of the dynamic environment provided that the following conditions hold:

- the algorithm is given a reasonable recovery time so that all types of changes in the problem may appear. This means that the algorithm is able to recover after any change, even if it needs a long time for it, and
- the global maximum fitness is restricted to a small range of values.

Taking into account those conditions Morrison proposes another new measure that he called the *Collective Mean Fitness F_C*, which is *"a single value that is designed to provide an aggregate picture of an EA's performance, where the performance information has been collected over a representative sample of the fitness landscape dynamics. Collective fitness is defined as the mean best-of-generation values, averaged over a sufficient number of generations, G', required to expose the EA to a representative sample of all possible landscape dynamics, further averaged over multiple runs"*. Mathematically this can be expressed as:

$$F_C = \frac{\sum_{m=1}^{M} \left(\frac{\sum_{g=1}^{G'} (F_{BG})}{G'} \right)}{M} \tag{2.2}$$

The collective mean fitness F_C will approach the total mean fitness F_T whenever the optimization process has a sufficient large exposure to the dynamics of changes. *Sufficient* means large enough to provide a representative sample of the fitness dynamics and to allow the algorithm to stabilize the running average best-of-generation fitness value.

Therefore, to be able to use F_C, the number of runs of the algorithm has to be large enough to provide a representative sample of the problem dynamics. Finding the necessary number of runs can be a difficult task in problems with unknown dynamics. In those cases, the number of runs has to be guessed by experimenting with the given problem.

Weicker's Measures.

Karsten Weicker proposes (Weicker, 2002) measures for what he describes as the three different aspects that have to be taken into account when analysing and comparing algorithms for dynamic problems. First of all, there is an *accuracy* measure, first introduced in Feng, Brune, Chan, Chowdhury, Kuek, and Li (1997) as a performance measure for stationary problems. *accuracy* should measure the closeness of the current best found solution to the actual best solution. It takes values between 0 and 1, where 1 is the best accuracy value. It is defined as:

$$accuracy_{F,EA}^{(t)} = \frac{F(best_{EA}^{(t)}) - min_F^{(t)}}{max_F^{(t)} - min_F^{(t)}} \tag{2.3}$$

where $best_{EA}^{(t)}$ is the best solution found by an evolutionary algorithm (EA) in the population at time t. The maximum and minimum fitness values in the search space are represented by $max_F^{(t)}$ and $min_F^{(t)}$, respectively. F is the fitness function of the problem.

Weicker also stated that *stability* is an important issue in the context of dynamic optimization. A dynamic algorithm is called stable if changes in the environment do not affect the optimization accuracy severely. Hence, a definition for a *stability* measure was given as:

$$stab_{F,EA}^{(t)} = \max\left\{0, accuracy_{F,EA}^{(t)} - accuracy_{F,EA}^{(t-1)}\right\} \tag{2.4}$$

and takes values from 0 to 1. In this case, a value close to 0 means high stability.

A third aspect of interest in dynamic problems is the ability of an algorithm to react to changes. Weicker proposes to check whether an algorithm has *ε-reactivity* at time t using the next equation:

$$react^{(t)}_{F,EA,\varepsilon} = \min \left\{ \left\{ t' - t \middle| t < t' \leq maxgen, t' \in \mathbb{N}, \frac{accuracy^{(t')}_{F,EA}}{accuracy^{(t)}_{F,EA}} \geq (1-\varepsilon) \right\} \bigcup \{maxgen - t\} \right\} \tag{2.5}$$

where *maxgen* is the number of generations. The measure $react^{(t)}_{F,EA,\varepsilon}$ evaluates how much time Δt took the algorithm to achieve a desired accuracy threshold.

Different Proposals for Accuracy.

Since it is difficult to know which is the best achievable value in a dynamic problem, Weicker points out that an average of several generations should be used instead. This approach was already proposed in Mori, Kita, and Nishikawa (1996). That averaged measure turns out to be very similar to that proposed in Trojanowski and Michalewicz (1999) but the normalization of the average does not depend on the worst fitness value. Additionally, in Mori et al. (1998) more emphasis was put on the detection of the optimum by proposing a different mechanism to weight the fitness values. Another proposal was made by Hadad and Eick (1997) where they include the squared error of the best fitness value.

Weicker also summarizes different proposals for *accuracy* in problems where the global optimum is unknown, offering the following options to be used instead of the $best^{(t)}_{EA}$ value in (2.3):

$$currentBest^{(t)}_{F,EA} = \max \left\{ F(\omega) \mid \omega \in P^{(t)}_{EA} \right\} \tag{2.6}$$

$$currentBestOffline^{(t)}_{F,EA} = \max_{1 \leq t' \leq t} \left\{ currentBest^{(t')}_{F,EA} \right\} \tag{2.7}$$

$$currentAverage^{(t)}_{F,EA} = \frac{\sum\limits_{\omega \in P^{(t)}_{EA}} (F(\omega))}{|P^{(t)}_{EA}|} \tag{2.8}$$

where $P^{(t)}_{EA}$ is the population of the algorithm at time t.

However these measures could be applied to any algorithm used to solve a single objective dynamic problem, Weicker uses the *EA* subscript to indicate that they are applied to evolutionary algorithms. The author of this thesis has tried to respect to the maximum the terminology used by every author. Because of this, the *EA* subscript has been kept in all the definitions that have been taken from Weicker (2002).

According to Weicker (2002), most researchers use the best fitness value $currentBest^{(t)}_{F,EA}$, while the measure $currentBestOffline^{(t)}_{F,EA}$ is not suitable because it compares values from different generations where the problem may behave in a very different way (Grefenstette, 1999). Additionally, Branke (1999a) uses a mixed approach where only those values from generations without changes in the environment are compared. This means that the measure can be only obtained *offline* because it requires a global knowledge of the problem.

Another approach, based on the assumption that the best fitness value will not change much within a small number of generations, is employed to measure the *accuracy* without actually knowing the best fitness. Hence, a window is defined inside the time span of the problem. Therefore, a proposal is given to substitute the expression $accuracy^{(t)}_{F,EA}$ (2.3) with a suitable version of the *accuracy* measure whenever it should be defined inside a window of length W:

$$\begin{aligned} windowAcc^{(t)}_{F,EA,W} &= \max\left\{ \frac{F(\omega) - windowWorst}{windowBest - windowWorst} \mid \omega \in P^{(t)}_{EA} \right\} \\ \text{with} \quad windowBest &= \max\left\{ F(\omega) \mid \omega \in P^{(t')}_{EA}, t - W \leq t' \leq t \right\}, \\ \text{and} \quad windowWorst &= \min\left\{ F(\omega) \mid \omega \in P^{(t')}_{EA}, t - W \leq t' \leq t \right\}. \end{aligned} \tag{2.9}$$

Finally, as an alternative to the fitness based performance measures, *genotype* or *phenotype* based measures can also be used to give an approximated value of the *accuracy*. Weicker notes that these measures require full global knowledge of the position of the current optimum and gives two variants. The first proposal (Weicker and Weicker, 1999) uses the minimal distance of the individuals in the population to the current optimum $\omega^* \in \Omega$ (where Ω represents the search space) giving the expression:

$$bestDist^{(t)}_{F,EA} = \max\left\{ \frac{maxdist - d(\omega^*, \omega)}{maxdist} \mid \omega \in P^{(t)}_{EA} \right\} \tag{2.10}$$

where *maxdist* is the maximum distance between two solutions in Ω.

On the other hand, the second approach (Salomon and Eggenberger, 1998) is based on the distance from ω^* to the mass centre or centroid of the population ω_{center} and is obtained by:

$$centerDist^{(t)}_{F,EA} = \frac{maxdist - d(\omega^*, \omega_{center})}{maxdist}. \tag{2.11}$$

Yu's Measures.

We end the review of performance measures for single objective problems with the proposed measures found in Yu, Tang, Chen, and Yao (2009). In it, the authors suggest using three kinds of measures. The first one is *Performance* that measures how well the system can do. In order to define their *Performance* measure, they use the *Collective Mean Fitness* (2.2). In addition, they introduce the overall *Average performance* of an algorithm defined as:

$$\overline{F}_{Avg} = \frac{\sum\limits_{i=1}^{G}\left(\frac{1}{N}\sum\limits_{j=1}^{N} F_{Avg_{i,j}}\right)}{G}, \tag{2.12}$$

where G is the total number of generations for a run, N is the total number of runs, and $F_{Avg_{i,j}}$ is the average fitness of the population of generation i of run j.

They also propose a measure for the *Robustness* of the solutions. It features the persistence of the solutions' fitness while taking into account the system and control influences (Bosman, 2005). System influence is the response of the dynamic system to the changes that it experiences over time, while the control influence is the response of the system at a given time to the decision made by the algorithm in the past. Hence, Yu et al define two new measures: the *best robustness* of generation i and the *average robustness* of generation i. They are defined as:

$$\overline{R}_{Best_i} = \frac{\sum\limits_{j=1}^{N} R_{Best_{i,j}}}{N}, \tag{2.13}$$

and

$$\overline{R}_{Avg_i} = \frac{\sum\limits_{j=1}^{N} R_{Avg_{i,j}}}{N}, \tag{2.14}$$

respectively. $R_{Best_{i,j}}$ and $R_{Avg_{i,j}}$are the *best robustness* and *average robustness* of generation i of run j, respectively, which are defined as:

$$R_{Best_{i,j}} = \begin{cases} 1, & \text{if } \frac{F_{BOG_{i,j}}}{F_{BOG_{i-1,j}}} > 1 \\ \frac{F_{BOG_{i,j}}}{F_{BOG_{i-1,j}}}, & \text{otherwise} \end{cases} \tag{2.15}$$

and

$$R_{Avg_{i,j}} = \begin{cases} 1, & \text{if } \frac{F_{Avg_{i,j}}}{F_{Avg_{i-1,j}}} > 1 \\ \frac{F_{Avg_{i,j}}}{F_{Avg_{i-1,j}}}, & \text{otherwise} \end{cases}, \tag{2.16}$$

respectively, where *BOG* means best of generation. According to the authors, higher *Robustness* levels indicate more persistent fitness levels. This measure is quite similar to the way that *stability* will be defined for multi-objective problems in the next section.

Finally, the authors give a *diversity* measure of how different are the individuals of the population but we must remember that it is used for single objective problems where *diversity* is not an essential desired feature of the optimization algorithm as it happens to be for multi-objective optimization algorithms. Hence, the *diversity* measure of generation i is defined as

$$\overline{Div_i} = \frac{1}{N} \sum_{j=1}^{N} Div_{i,j} \tag{2.17}$$

where $Div_{i,j}$ is the diversity of generation i of run j. $Div_{i,j}$ must be defined specifically for the problem which is under study (Yu et al., 2009).

2.3.2 Multi-objective Performance Measures

Unlike performance measures for single objective problems, performance measures for multi-objective problems have drawn little attention in the literature. Indeed, very few proposals, which are discussed in the rest of this section, can be found in it.

The problem shown by all the measures proposed until now and the corresponding literature about them is that these measures were designed only for dynamic single objective problems. As it has been shown in the previous section, the main difficulty to design good performance measures lies in dealing with those problems which have unknown Pareto fronts.

In addition to that hurdle, the fundamental difference between single and multi-objective dynamic problems, where there is a set of Pareto optimal solutions, makes necessary to define and adapt those measures for those problems. Then, the researcher, trying to find suitable performance measures for dynamic multi-objective problems, faces two obstacles that are essentially orthogonal or independent:

- the multi-objective character of the problem, meaning that performance measures have to evaluate the algorithm after finding a set which has to contain a large enough number of different solutions, and

- the fact that the location of the real Pareto front is very unlikely to be known at any given instant time of the execution of the algorithm.

In Li, Branke, and Kirley (2007), some measures have been proposed. The first one, called *reverse generational distance* or $rGD(t)$, is an adaptation of the generational distance G proposed in van Veldhuizen (1999). For completeness' sake, G is reproduced here:

$$G = \frac{\left(\sum_{i=1}^{n} d_i^p\right)^{1/p}}{n}, \tag{2.18}$$

where n is the population size of the found approximation set, $p = 2$, and d_i is the Euclidean distance from the i-th found solution to the nearest solution in the real Pareto front. Thus, G is the average distance of the solutions in the approximation set to those solutions representing the real Pareto front of the algorithm.

$rGD(t)$ is called *reversed* generational distance because the matching of the solutions from the current approximate Pareto front to the real Pareto front is done from the latter to the former, in contrast to the definition of G (van Veldhuizen, 1999). Moreover, time is taken into consideration as the measure depends on it. G was intended only for stationary multi-objective problems, while $rGD(t)$, defined in what follows, is for dynamic ones:

$$\begin{aligned} rGD(t) &= \frac{\sum_{i=1}^{|\mathcal{P}^*(t)|} d_i}{|\mathcal{P}^*(t)|} \\ \text{where } d_i &= \min_{i=1}^{|\mathcal{Q}(t)|} \left\{ \sqrt{\sum_{j=1}^{M} \left(f_j^{*(i)} - f_j^{(k)} \right)^2} \right\}, \end{aligned} \tag{2.19}$$

with $\mathcal{P}^*(t)$ being the real Pareto front, $\mathcal{Q}(t)$ an approximation set at time t and $f_j^{(k)}$ the j-th objective function value of the k-th member of $\mathcal{Q}(t)$. The smaller the $rGD(t)$, the better the approximation set represents the real Pareto front. Thus, $rGD(t)$ lies in the category of *accuracy* performance measures.

The drawback of $rGD(t)$ is that the location of the real Pareto front $\mathcal{P}^*(t)$ must be known at any time. Nevertheless, it could represent a good adaptation to multi-objective problems of Weicker's *accuracy* (2.3).

In Li et al. (2007), it is pointed out that there is a similar measure, $D1_R$ (Czyzak and Jaszkiewicz, 1998), that also measures the average distance from sampling points on $\mathcal{P}^*(t)$ to the nearest point on $\mathcal{Q}(t)$. It is defined as follows:

$$D1_R(A, \Lambda) = \frac{\sum\limits_{r \in R} \min\limits_{z \in A} \{d(\mathbf{r}, \mathbf{z})\}}{|R|}, \tag{2.20}$$

where A and R are sets equivalent to $\mathcal{Q}(t)$ and $\mathcal{P}^*(t)$, respectively, $d(\mathbf{r}, \mathbf{z}) = \max\left\{\lambda_j(r_j - z_j)\right\}$ and $\Lambda = |\lambda_1, \ldots, \lambda_J|, \lambda_j = 1/R_j, j = 1, \ldots, J$ with R_j being the range of objective j in set R.

Li et al demonstrate in their paper that the measure $D1_R$ behaves unexpectedly with respect to some cases, while $rGD(t)$ does not.

Another drawback of $rGD(t)$ is that not only a distribution of the real Pareto front $\mathcal{P}^*(t)$ must be known beforehand, but it also must have an adequate distribution of sampling points of the whole Pareto front.

They also propose to use the *hypervolume ratio* $HVR(t)$, first introduced in Deb (2001), for dynamic multi-objective problems. This measure is identical to our accuracy measure (Cámara et al., 2007a) which is introduced in the next section.

Finally, they introduce a measure called *Collective Mean Error CME*. It is quite similar to the *collective mean fitness* proposed by Morrison (2.2). This *CME* measure is calculated by averaging the $rGD(t)$ values over a full run of the algorithm and it is expressed by:

$$CME_{rGD} = \frac{\sum\limits_{t=1}^{T} rGD(t)}{T}, \tag{2.21}$$

where T is the total number of iterations of a run. They also give a similar definition of the *CME* measure (2.21) but in terms of $HVR(t)$:

$$CME_{HVR(t)} = \frac{\sum_{t=1}^{T} HVR(t)}{T}, \tag{2.22}$$

it is impossible that the whole dynamic behaviour of one full run of the problem can be collected in only just one measure. This resembles the idea that it is not possible to represent a random data set with only a representative datum as the mean or the standard deviation without knowing the distribution which follow the data from the set. Because of this, the author of this dissertation greatly discourages the use of such collective measures since a dynamic problem is meant to change over the time, and every generation involves a set with many solutions. Therefore, such collective information for these problems must be gathered in every instant of time of the whole run of the algorithm.

Table 2.3, found at the end of this chapter due to space limitations, summarizes all the reviewed measures and the corresponding features they are related with according to the classifications seen in Section 2.1.1 and summed up in Figure 2.1. It can be seen that almost all the measures are *offline* and only two of them can be calculated *online*. It also shows that most of the measures are for *diversity*, *accuracy* and *stability*, and there is only one more measure which is for the *reaction time*. This also means that no measures have been offered for the *throughput* and the *time* feature. The latter is understandable because a measure for *time* is just a count of the number of seconds that took the algorithm to complete its assigned number of iterations. Moreover, it should be pointed out that all the measures require either *partial* or *full information* in order to be calculated.

2.4 Proposed Measures

When taking into account measures for dynamic multi-objective problems it is important to make a clear difference between those problems in which the current real Pareto front is known at every time of the algorithm execution and those in which the real Pareto front is rather unknown. The latter is the usual case in real world problems, even in many of the test cases suggested for researching purposes (Farina et al., 2004).

All of the already indicated measures that have been previously proposed are only applicable either to single objective problems or to those multi-objective problems in which the real Pareto fronts, $F_{real}(t)$, are known at any time. In this section, we contribute to solve this problem with a set of new measures aimed at the dynamic character of DMO problems.

Before describing our proposed measures there is a notation issue that has to be considered, because there is not a common and unified way to describe the problems in the field of multi-objective optimization. In the rest of this thesis, we will use a notation to designate the different aspects of each problem that is strongly influenced from that found in Farina et al. (2004). The terms *decision space* and *objective space* refer, respectively, to the spaces of the search variables and of the objective functions of that search variables. Also, we call the set of non-dominated solutions found at time t as *approximate Pareto optimal solutions* at time t, which is divided into the *approximate decision space* at time t, $\mathbb{S}_P(t)$, and the *approximate objective space* or *approximate Pareto front* at time t, $\mathbb{F}_P(t)$. The real Pareto front, that could be known or not at time t, is denoted as $F_{real}(t)$ and is always in the objective space. Finally, $V(t)$ is the hypervolume value of the approximation set at time t. If different approximation sets are available, a subscript will be used to indicate the one used.

2.4.1 Measures When the Fronts are Known.

Figure 2.2 describes a possible scenario for a dynamic multi-objective minimization problem. In this scenario, solutions from the current approximation to the Pareto front are represented by black dots. This front has a *hypervolume* value calculated from the darker shaded area. This *hypervolume* value of the current approximation is between the *hypervolume* values corresponding to the minimum and maximum approximate Pareto fronts found so far (in dotted lines), which are represented by $V^{min}(t)$ and $V^{max}(t)$, respectively. Finally, as the problem is dynamic, the real Pareto front, which is shown in the Figure by the solid line, has moved and thus the current maximum *hypervolume* value found $V^{max}(t)$ is bigger than the current *hypervolume* value $V(t)$ which corresponds to the current approximation set found. Moreover, if the real Pareto fronts were unknown in the previous steps of the algorithm, $V^{min}(t)$ and

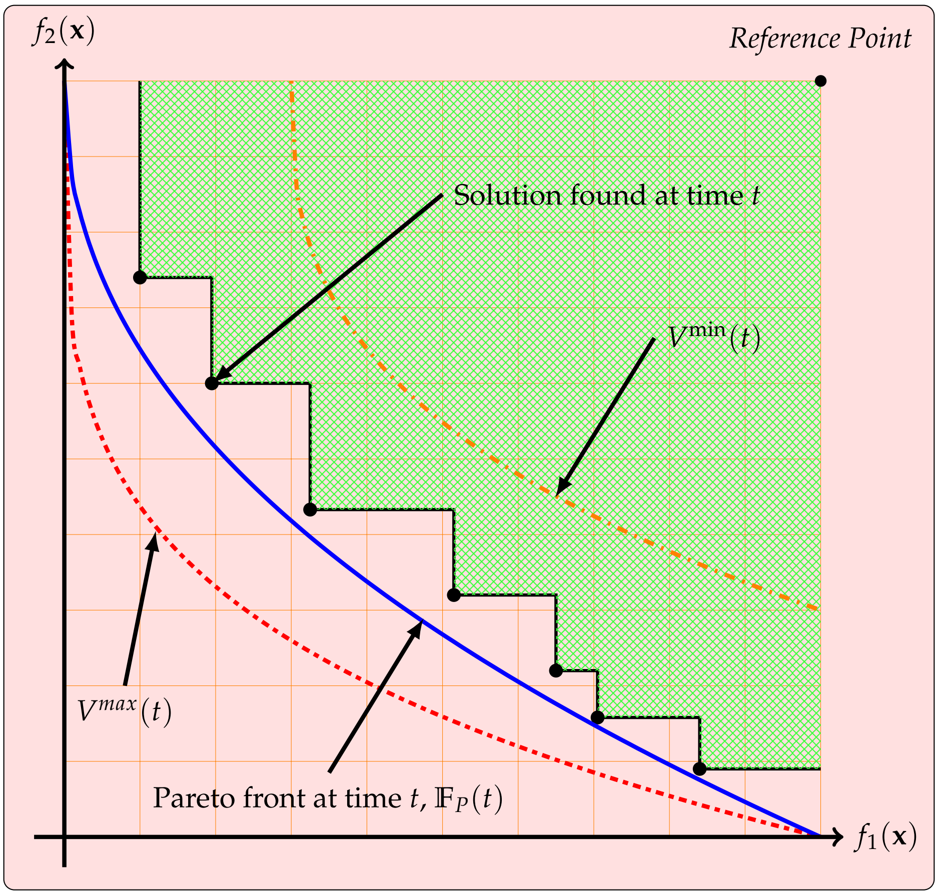

Figure 2.2: Description of approximations to the Pareto front at different times.

$V^{max}(t)$ are only guesses based on the approximation set found earlier in the run of the algorithm and represent the maximum and minimum values found for the *hypervolume* measure.

The dynamic nature of the real Pareto fronts makes necessary to come up with new performance measures able to cope with the changing characteristics of these problems.

Accuracy. We proposed a modification of *accuracy* (2.3) (Cámara et al., 2007a) to make it calculable from the *hypervolume* values of the current approximation set and the real Pareto front. This proposed measure proved to be identical to $HVR(t)$ (Li et al., 2007).

Due to the dynamic nature of the problem, we require to elaborate the *accuracy* concept a little bit more in order to define a suitable measure for these dynamic applications. The difficulty appears because there are minimizing and maximizing problems. In other words, a minimizing problem will have hypervolume values that grow provided that the problem conditions have not changed and the algorithm is improving the solutions found. On the other hand, given that we have a maximization problem in a stationary state, the hypervolume values will decrease. In those cases where some of the objective functions are minimizing and others are maximizing at the same time, the functions have to be converted to either all minimizing or either all maximizing. Therefore, an *accuracy* measure that is computed as the rate between the hypervolume of the current approximate Pareto front and the current real Pareto front could lead to values above one.

Consequently, we have to define a *dual accuracy measure* that allows different ways to calculate the *accuracy* values according to the nature of the problem. Thus, we propose the following:

$$acc_{minimizing}(t) = \frac{HV(F_p(t))}{HV(F_{real}(t))} = \frac{V(t)}{V_{real}(t)} \tag{2.23}$$

$$acc_{maximizing}(t) = \frac{HV(F_{real}(t))}{HV(F_p(t))} = \frac{V_{real}(t)}{V(t)} \tag{2.24}$$

$$acc(t) = \begin{cases} acc_{maximizing}(t) & \text{for a maximization problem} \\ acc_{minimizing}(t) & \text{for a minimization problem} \end{cases} \tag{2.25}$$

An alternative *accuracy* measure could be used instead of the above $acc(t)$ (2.25) by replacing the rate between the current approximate Pareto front and the real one with a subtraction between them:

$$acc_{alternative}(t) = |V_{real}(t) - V(t)|. \tag{2.26}$$

One of the advantages of (2.26) is that it is independent of the dichotomy of the maximizing/minimizing nature of the problem at time t. On the other hand, it takes unbounded positive real values with 0 as the best accuracy.

A more specific accuracy measure could be elaborated to be used with problems where a detailed knowledge of the underlying structure of the decision and objective space could make possible to replace the *hypervolume* value measures with any of the quality indicators or attainment functions compiled in Knowles et al. (2006).

Given a suitable *accuracy* measure for multi-objective problems, the other two measures are easy to define.

Stability. In this case, we have modified Weicker's *stability* measure so that it produces the value 1 when the current accuracy has worsened in comparison to the previous one. Thus, we define it as:

$$stb(t) = \begin{cases} stb_0(t) & \text{if } stb_0(t) \geq 0 \\ 1 & \text{otherwise} \end{cases} \qquad \text{with } stb_0(t) = acc(t) - acc(t-1). \tag{2.27}$$

With this new definition $stb(t)$ takes values between 0 and 1, being 0 the best *stability* and 1 the worst one. Sometimes, Weicker's *stability* measure (2.4) is useless, because it represents the best and the worst values achieved by an algorithm by the value 0 at the same time. Although it is arguable that a null value for the best *stability* is very difficult to obtain, it is not impossible to achieve it. This way, it is not possible to tell whether a null value in (2.4) is indicating that the algorithm performed really well or badly.

Reaction Time. This measure allows us to know how much time takes the algorithm to recover its *accuracy* level when a change occurs in the problem. It is based on Weicker's *react* measure (2.5) but we have updated it by using our *accuracy* measure (2.25). After doing so, the new *reaction time* is expressed as:

$$reac_{\varepsilon}(t) = \min\left\{ \left\{ t' - t \,|\, t < t' \leq maxgen, t' \in \mathbb{N}, \frac{acc(t')}{acc(t)} \geq (1 - \varepsilon) \right\} \bigcup \{maxgen - t\} \right\}. \tag{2.28}$$

In (2.28), ε is a small positive real number used as a threshold for comparing different accuracy values. An alternative way to define the *reaction time*, $react_{alt,\varepsilon}$, is given by the following expression:

$$reac_{alt,\varepsilon}(t) = \min\left\{ \{ t' - t \,|\, t < t' \leq maxgen, t' \in \mathbb{N}, acc(t') - acc(t) \geq -\varepsilon \} \bigcup \{maxgen - t\} \right\}. \tag{2.29}$$

As it can be seen, the difference between (2.28) and (2.29) lies in the way the *accuracy*, $acc(t)$, and the ε values are employed.

2.4.2 Measures When the Fronts are Unknown.

As stated before, all the proposed measures given so far for dynamic problems rely on the knowledge of where the real Pareto fronts lie (Li et al., 2007; Cámara et al., 2007a). This occurs only in test cases specifically devised for evaluating algorithms. Because of this, it is mandatory to pay attention to new ways of redefining the measures described in the previous subsection in order to deal with real problems where the location of the real Pareto fronts is unknown. However, the only measure that has to be adapted is $acc(t)$ (2.25) as the other two, $stb(t)$ (2.27) and $reac_{\varepsilon}(t)$ (2.28), rely on the knowledge of Pareto fronts only through $acc(t)$ (2.25).

Nevertheless, we note again that as we are interested in an offline measure we can exploit the knowledge of all the approximate Pareto fronts, not only the past ones, but also those who came after the time instant which we are studying.

Thus, in order to improve our accuracy definition for those cases in which the real Pareto fronts are unknown we need to replace the *hypervolume* value of the real Pareto fronts by other suitable quantities. These quantities could be the maximum and minimum *hypervolume* values over the time. However, if the Pareto fronts of the problem objective space change, those absolute maximum and minimum measures could be far from the current real Pareto front. Because of this, *accuracy* is considered a measure of the current approximation set in comparison only with the nearby approximation set found, both in the past and in the future. This is the concept of *accuracy* within a window or offset which was already mentioned in Weicker (2002).

A *window* is a period of time in which the problem becomes stationary, or put in other words, a span of time in which the problem does not show clear signals of changes in the approximation sets that have been found for those times. A window marks a phase of the problem. Each phase is characterized by the moment in which a change has been made, the phase starting point, and by the duration or length of the phase which is given in time units.

The window length should not be a constant given by the researcher but a variable that it is calculated before applying the performance measures to the collected data. If the problem at hand changes with a fixed frequency, this window length would turn out to be equal to the inverse of that frequency. But in order to widen the set of problems to be able to analyse, the measure under study must be able to cope with variable frequencies.

To calculate the lengths of all the phases we propose a procedure described in Algorithm 2.1, where the *if* in line 6 can be changed to other conditions which may be useful to detect changes in the fronts.

Therefore, this improved measure has two parts. Firstly, the windows or phases are detected and the lengths corresponding to each phase are calculated with Algorithm 2.1. Afterwards, *accuracy* values are calculated at every time step using the relative minimal or maximal *hypervolume* values within that phase.

Once the lengths have been obtained the *accuracy* is calculated for

Algorithm 2.1: Calculation of lengths.

Input: A set of N hypervolume values for the approximate Pareto fronts.

Output: A set S of the lengths of each of the phases.

```
1  begin
2      for $i = 2$ to $N$ do
3          $\Delta HV_i = HV_i - HV_{i-1}$
4      $length = 1$
5      for $i = 2$ to $N$ do
6          if $\Delta HV_i \geq |\Delta HV_{i-1} + \Delta HV_{i+1}|$ then
7              $S \longleftarrow S \cup length$
8              $length = 1$
9          else
10             $length = length + 1$
11 end
```

every approximate Pareto front found with $acc_{unk}(t)$ (2.30), where unk makes emphasis on the fact that the measure is applied to problems with unknown real Pareto fronts. Thus, $acc_{unk}(t)$ is defined with the help of (2.31) and (2.32), giving:

$$acc_{unk}(t) = \begin{cases} acc_{unk}^{maximizing}(t) & \text{if the problem is maximizing} \\ acc_{unk}^{minimizing}(t) & \text{if the problem is minimizing} \end{cases} \tag{2.30}$$

$$acc_{unk}^{maximizing}(t) = \frac{HV_{min}(\mathcal{Q}(t))}{HV(F_p(t))} = \frac{HV(F_p(\min\{\mathcal{Q}(t)\}))}{HV(F_p(t))} = \frac{V_{min}^{\mathcal{Q}(t)}}{V(t)} \tag{2.31}$$

$$acc_{unk}^{minimizing}(t) = \frac{HV(F_p(t))}{HV_{max}(\mathcal{Q}(t))} = \frac{HV(F_p(t))}{HV(F_p(\max\{\mathcal{Q}(t)\}))} = \frac{V(t)}{V_{max}^{\mathcal{Q}(t)}} \tag{2.32}$$

where $\mathcal{Q}(t)$ is a set containing the time values for the window in which t takes place, i.e., the surrounding past and future values of t in which the approximate Pareto fronts have not suffered a noticeable change in its *hypervolume* values, according to algorithm 2.1. The cardinality of each set $\mathcal{Q}(t)$ equals the duration of the phase that $\mathcal{Q}(t)$ represents.

In Algorithm 2.1, and in (2.31) and (2.32), the *hypervolume* measures of the approximate Pareto fronts that evaluate $acc_{unk}(t)$ may be changed to other equivalent measures such as those described in subsection 2.4.1.

Figure 2.3 shows how Algorithm 2.1 can be used to detect a change in the problem. In Figure 2.3, the optimal Pareto fronts for FDA3-mod (Eq. 2.35) are shown when τ equals $4, 5$ and 10. It can be seen that the change in the *hypervolume* values of the fronts is big enough to allow Algorithm 2.1 to detect a new phase in the problem. In this example, the optimal Pareto fronts are shown in three different values of τ.

Because we are showing the inner workings with this example, the optimal Pareto front is found in only one step of τ, instead of needing some more steps before the algorithm converges to the new Pareto front, which would be the usual procedure for real problems.

It is important to note that even when the Pareto fronts are obtained by an optimization algorithm, if a change has been produced, the new Pareto front will be farther than the earlier front was. Thus, the detection of a change would be even easier to perform. Moreover, as Algorithm 2.1 uses the absolute value of ΔHV_i, the detection mechanism works even when the new Pareto front is going towards the origin after a change instead of going outwards from the origin. It can be seen in the plot that when τ changes from 4 to 5, the problem conditions also change, and the Pareto front is shifted. This happens only in the moment that τ becomes 5 and not for any of the values between $0 \leq \tau < 5$.

Although Figure 2.3 represents only optimal Pareto fronts, the proposed procedure is able to track problems with unknown Pareto fronts, and results on this will be shown in Section 4.5.

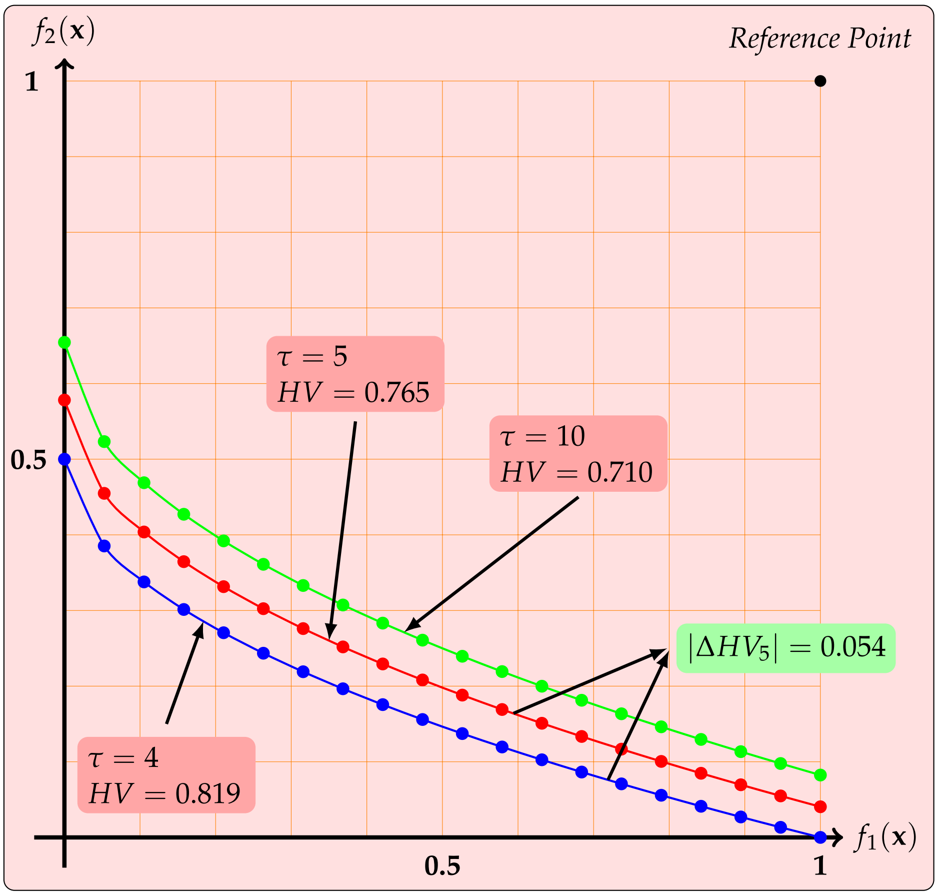

Figure 2.3: Illustrative use of the length detection.

2.5 Test Cases for Dynamic Multi-objective Optimization

This section reviews the five test cases for DMO that are used along this thesis. They stem from those proposed by Farina et al (Farina et al., 2004), because these five functions have become the standard test suite for researchers in the field of DMO. Nevertheless, we propose modified versions of the FDA2 and FDA3 original functions in order to overcome some underlying problems in the original functions. Other modifications for FDA2 have been proposed in Deb et al. (2007) and Mehnen, Wagner, and Rudolph (2006). See Cámara, Ortega, and de Toro (2008a) and Cámara (2007) for further details about our decision to bring a third alternative for FDA2. For FDA3 no modification apart from the one suggested by us has been proposed in the literature.

In order to create a set of functions that reflect the dynamic nature of DMO, the five FDA test cases depend on a periodic function that affects the overall output of the test case as the time advances. This periodic function simulates the concept of time by using sinus or cosine functions together with some parameters that adapt the periodic function to the test case where it is being used.

The parameters that control this periodic function are four:

- τ is the generation counter. It is the time value at which the FDA test case is being evaluated, and it is a input parameter to the test case together with the decision variables vector $\mathbf{x}$.
- τ_T defines the number of discrete time intervals where the test case remains non-stationary. In other words, for τ_T values of τ the test case does not change.
- t is an internal time function that depends on the current values of n_t, τ and τ_T.
- n_t is the number of distinct steps in t.

In this thesis, as it has been suggested in Farina et al. (2004), the values used are always the following:

- $\tau_T = 5$.
- $n_t = 10$.

There are three two-dimensional functions: FDA1, FDA2-mod and FDA3-mod, and two many-dimensional functions: FDA4 and FDA5.

In what follows, every FDA test case used in this thesis is described.

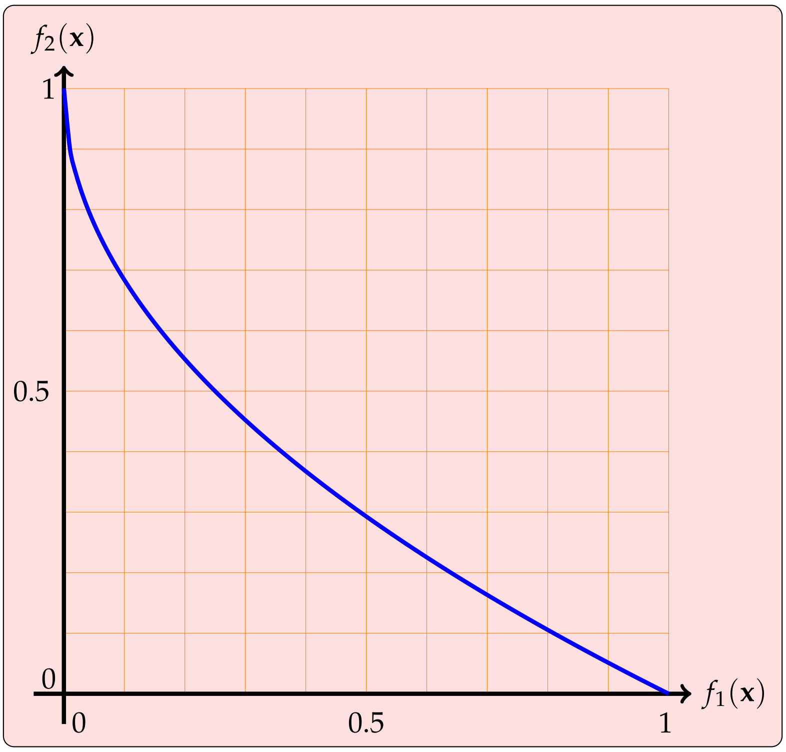

Figure 2.4: Pareto solutions for FDA1 for any value of τ

2.5.1 FDA1.

In the first test function, FDA1 (Eq. 2.33), the Pareto front, $F_p(\tau)$, remains equal for all values of τ, while the values of the decision variables to the corresponding front, $S_p(\tau)$, do change. The solution sets for FDA1 are $|X_I| = 1$ and $|X_{II}| = 19$. In Figure 2.4 it can be seen the plot of the FDA1 function for the any value of τ.

$$\textbf{FDA1} = \begin{cases} f_1(\mathbf{x}_I) = x_1 \\ f_2(\mathbf{x}) = g(\mathbf{x}_{II})h(f_1, g) \\ g(\mathbf{x}_{II}) = 1 + \sum\limits_{x_i \in \mathbf{x}_{II}} \Big(x_i - G(t)\Big)^2 \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \\ G(t) = \sin(0.5\pi t)\ , \qquad t = \frac{1}{n_t} \lfloor \frac{\tau}{\tau_T} \rfloor \\ \mathbf{x}_I = (x_1) \in [0, 1], \qquad \mathbf{x}_{II} = (x_2, \ldots, x_n) \in [-1, 1] \end{cases} \tag{2.33}$$

2.5.2 FDA2-mod.

We call our modified versions of the original FDA2 and FDA3 *FDA2-mod* and *FDA3-mod*, respectively. In FDA2-mod (Eq. 2.34), together with the values of the solutions, $S_p(\tau)$, also the corresponding Pareto front, $F_p(\tau)$, changes. For FDA2-mod, the solution sets are $|X_I| = 1$ and $|X_{II}| = |X_{III}| = 15$. We suggest a value of $z = 5$ in $H(t)$.

$$\textbf{FDA2-mod} = \begin{cases} f_1(\mathbf{x}_I) = x_1 \\ f_2(\mathbf{x}) = g(\mathbf{x}_{II})h(f_1, g) \\ g(\mathbf{x}_{II}) = 1 + \sum\limits_{x_i \in \mathbf{x}_{II}} x_i^2 \\ h(\mathbf{x}_{III}, f_1, g) = 1 - \left(\frac{f_1}{g}\right)^{\left(H(t) + \sum\limits_{x_i \in \mathbf{x}_{III}} (x_i - H(t)/2)^2\right)} \\ H(t) = z^{-\cos(\pi t/4)}\ ; \qquad t = \frac{1}{n_t} \lfloor \frac{\tau}{\tau_T} \rfloor \\ \mathbf{x}_I = (x_1) \in [0, 1], \qquad \mathbf{x}_{II}, \mathbf{x}_{III} \in [-1, 1] \end{cases} \tag{2.34}$$

With respect to FDA2-mod, Figure 2.5 shows the Pareto fronts that presents the function from $\tau = 5$ to $\tau = 200$. These solutions reflect all the Pareto fronts. In FDA2-mod, when the generation τ crosses a τ_T generation border, the current Pareto front advances to the next one, and the values of the solution space change accordingly.

2.5.3 FDA3-mod.

The definition of FDA3-mod is given in Eq. 2.35 and it uses the sets $|X_I| = 1$ and $|X_{II}| = 29$.

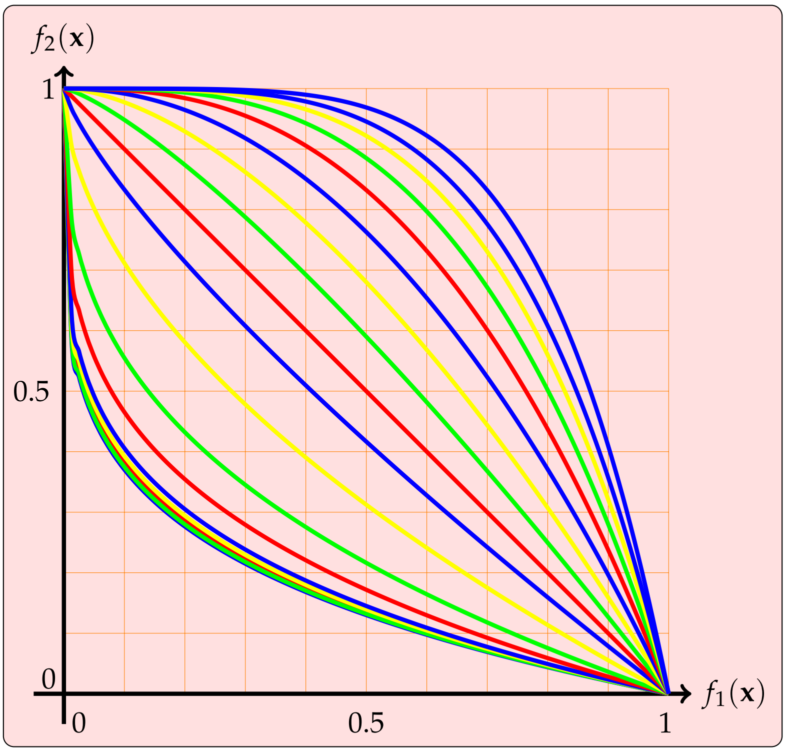

Figure 2.5: Pareto solutions for FDA2-mod for some values of τ ranging from 5 to 200.

$$\textbf{FDA3-mod} = \begin{cases} f_1(\mathbf{x}_I) = x_1^{F(t)} \\ f_2(\mathbf{x}) = g(\mathbf{x}_{II})h(f_1, g) \\ g(\mathbf{x}_{II}) = 1 + G(t) + \sum\limits_{x_i \in \mathbf{x}_{II}} (x_i - G(t))^2 \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \\ G(t) = |\sin(0.5\pi t)| \\ F(t) = 10^{2\sin(0.5\pi t)} \qquad t = \frac{1}{n_t}\lfloor \frac{\tau}{\tau_T} \rfloor \\ \mathbf{x}_I = (x_1) \in [0,1], \qquad \mathbf{x}_{II} = (x_2, \dots, x_n) \in [-1,1] \end{cases} \tag{2.35}$$

In Figure 2.6, FDA3-mod it is plotted with the Pareto fronts obtained

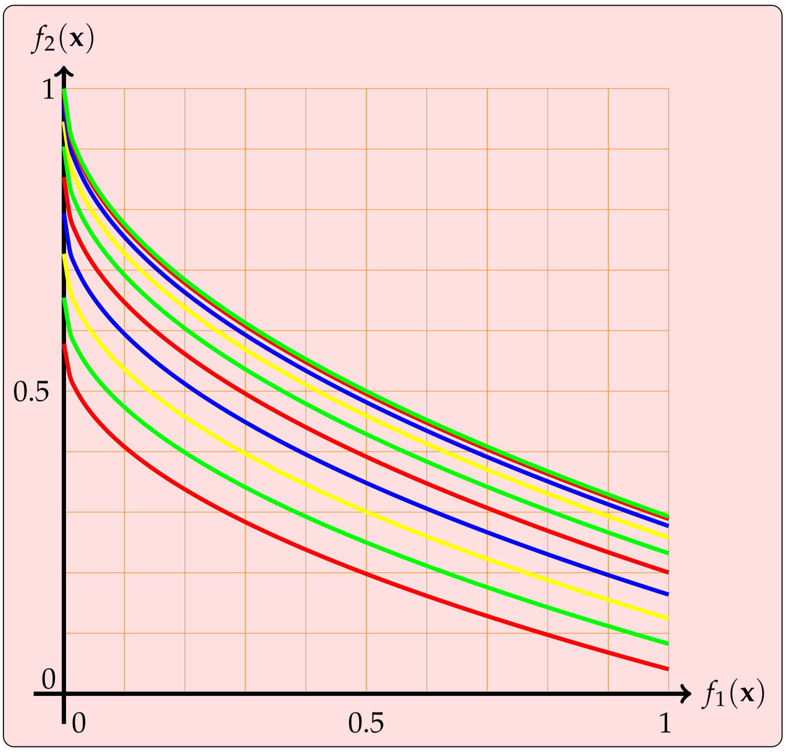

Figure 2.6: Pareto solutions for FDA3-mod from $\tau = 5$ to $\tau = 50$.

with τ ranging from 5 till 50.

2.5.4 FDA4.

FDA4 is a many dimensional function whose definition is given in 2.36. As it happened with FDA1, in FDA4 the Pareto front, $F_p(\tau)$, remains equal for all values of τ, while the values of the decision variables to the corresponding front, $S_p(\tau)$, do change with τ.

In order to get a three objective function, the value M is fixed to 3.

In this case, the Pareto front is always an octave of the sphere of radius one. A plot of the part of the sphere that represents the Pareto front is shown in Figure 2.7. In this thesis, $n = M + 9$ or $n = 12$.

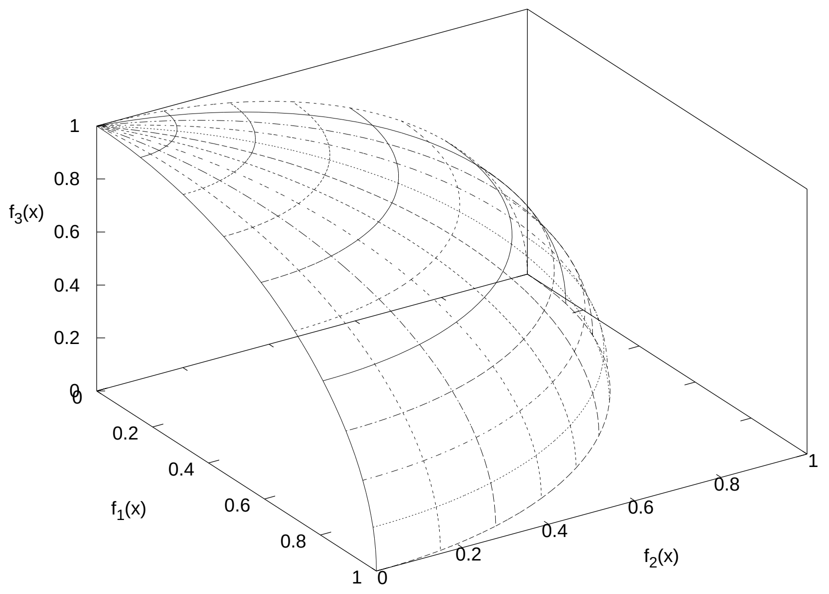

Figure 2.7: The spherical surface of radius one where the Pareto front lies for FDA4.

$$
\mathbf{FDA4} = \begin{cases}
f_1(\mathbf{x}) = (1 + g(\mathbf{x}_{II})) \prod_{i=1}^{M-1} \cos(\frac{x_i \pi}{2}) \\
f_k(\mathbf{x}) = (1 + g(\mathbf{x}_{II}))(\prod_{i=1}^{M-k} \cos(\frac{x_i \pi}{2})) \sin(\frac{x_{M-k+1}\pi}{2}) \text{ for } 2 \le k \le M-1 \\
f_M(\mathbf{x}) = (1 + g(\mathbf{x}_{II})) \sin(\frac{x_1 \pi}{2}) \\
g(\mathbf{x}_{II}) = \sum_{x_i \in \mathbf{X}_{II}} (x_i - G(t))^2 \\
G(t) = |\sin(\frac{t\pi}{2})|, \quad t = \frac{1}{n_t} \lfloor \frac{\tau}{\tau_T} \rfloor \\
\mathbf{x}_{II} = \{x_M, \ldots, x_n\}, \qquad x_i \in [0,1] \text{ for } 1 \le i \le n
\end{cases}
\tag{2.36}
$$

2.5.5 FDA5.

FDA5 is another many dimensional function. Once again, M is fixed to 3 to get a three objective function. The definition of FDA5 is given in 2.37. Contrary to what it happens for FDA4, the Pareto front, $F_p(\tau)$, for FDA5 changes with τ.

Although the Pareto front is still the spherical surface of an octave part of a sphere, the radius is not always one but it ranges from one to two. Figure 2.8 plots a cross section of different spherical surfaces of some FDA5 Pareto fronts. For the sake of simplicity, the plot shows only a small section of the whole surface for each Pareto front. Again, $n = 12$.

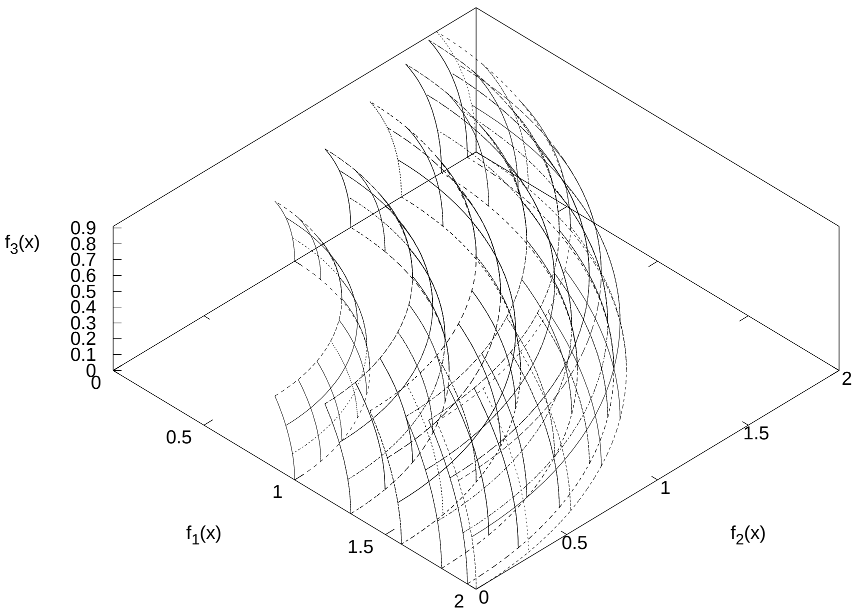

Figure 2.8: Cross section of the spherical surfaces for the Pareto fronts for FDA5 when $\tau = \{10, 20, 30, 40, 50\}$.

$$\textbf{FDA5} = \begin{cases} f_1(\mathbf{x}) = (1+g(\mathbf{x}_{II}))\prod_{i=1}^{M-1}\cos(\frac{y_i\pi}{2}) \\ f_k(\mathbf{x}) = (1+g(\mathbf{x}_{II}))(\prod_{i=1}^{M-k}\cos(\frac{y_i\pi}{2}))\sin(\frac{y_{M-k+1}\pi}{2}) \\ \qquad\qquad \text{for } 2 \leq k \leq M-1 \\ f_M(\mathbf{x}) = (1+g(\mathbf{x}_{II}))\sin(\frac{y_1\pi}{2}) \\ g(\mathbf{x}_{II}) = G(t) + \sum_{x_i\in\mathbf{X}_{II}}(x_i - G(t))^2 \\ y_i = x_i^{F(t)} \qquad \text{for } 1 \leq i \leq M-1 \\ F(t) = 1 + 100\sin^4(\frac{t\pi}{2}) \\ G(t) = |\sin(\frac{t\pi}{2})|, \quad t = \frac{1}{n_t}\lfloor\frac{\tau}{\tau_T}\rfloor \\ \mathbf{x}_{II} = \{x_M, \ldots, x_n\}, \qquad x_i \in [0,1] \text{ for } 1 \leq i \leq n \end{cases} \tag{2.37}$$

2.6 Summary

In this chapter, it has been shown how important it is for researchers to be able to use a good and complete suite of performance measures that can address the different aspects of any multi-objective dynamic optimization problem.

The main contributions of this chapter have been a classification of the performance measures and the introduction of new performance measures specifically proposed for multi-objective dynamic problems with or without known Pareto fronts. These contributions have been published in Cámara, Ortega, and de Toro (2009a,b); Cámara et al. (2010). Also the modifications proposed to the FDA test cases are important and they have been published in Cámara et al. (2010, 2008c,b).

Hopefully, these contributions get us, researchers, and our field of dynamic multi-objective optimization algorithms closer to the state that enjoy current-day mathematics after the findings made by Lebesgue, Hilbert and many other brilliant mathematicians of the past century, that made possible the development of most fields of mathematics (Doxiadis and Papadimitriou, 2009; Katz, 1998).

Fortunately, researchers are already paying more attention to these theoretical issues (Auger et al., 2009).

In the next chapter, it will be shown the modifications made to the algorithm SFGA (de Toro et al., 2004) that gave as a result a new evolutionary algorithm called SFGA2 (Cámara et al., 2008a,c). Results concerning the suitability of the new proposed algorithm SFGA2 will also be provided.

Table 2.3: Classification of the reviewed performance measures.

		Main Feature						Time		Knowledge		
										Independent	Dependent	
Measure	Expression	Diversity	Accuracy	Stability	Speed	Reaction	Yield	Offline	Online	Zero	Partial	Full
F_T	(2.1)		✓					✓				✓
F_C	(2.2)		✓					✓			✓	
$accuracy_{F,EA}^{(t)}$	(2.3)		✓					✓			✓	
$stab_{F,EA}^{(t)}$	(2.4)			✓				✓				✓
$react_{F,EA,\varepsilon}^{(t)}$	(2.5)					✓		✓			✓	
$windowAcc_{F,EA,W}^{(t)}$	(2.9)		✓					✓				✓
$bestDist_{F,EA}^{(t)}$	(2.10)		✓					✓			✓	
$centerDist_{F,EA}^{(t)}$	(2.11)		✓					✓			✓	
$\overline{F}_{Avg}$	(2.12)		✓					✓				✓
$\overline{R}_{Best_i}$	(2.13)			✓				✓				✓
$\overline{R}_{Avg_i}$	(2.14)			✓				✓				✓
$\overline{Div_i}$	(2.17)	✓						✓				✓
$rGD(t)$	(2.18)	✓	✓						✓		✓	
$D1_R(A, \Lambda)$	(2.20)	✓	✓						✓		✓	
CME_{rGD}	(2.21)	✓	✓					✓				✓
$CME_{HVR(t)}$	(2.22)	✓	✓					✓				✓

Table 2.4: Classification of the proposed new performance measures.

		Main Feature						Time		Knowledge		
										Independent	Dependent	
Measure	Expression	Diversity	Accuracy	Stability	Speed	Reaction	Yield	Offline	Online	Zero	Partial	Full
$acc(t)$	(2.25)		✓					✓			✓	
$acc_{alt}(t)$	(2.26)		✓					✓			✓	
$stab(t)$	(2.27)			✓				✓			✓	
$reac_{\epsilon}(t)$	(2.28)				✓			✓			✓	
$reac_{alt,\epsilon}(t)$	(2.29)				✓			✓			✓	
$acc_{unk}(t)$	(2.30)		✓					✓				✓

"There is nothing either good or bad, but thinking makes it so."
Hamlet, William Shakespeare

3 SFGA2: An improved version of SFGA

One of the main issues that is addressed in this thesis is the use of *evolutionary algorithms* (EAs) to solve dynamic multi-objective optimization problems. They are used within a framework, that is proposed in a later chapter, to solve DMO problems by using parallel processing. In order to obtain consistent results, the framework should be able to run different EAs so that a comparison between them could be done and conclusions about which algorithms behave better could be inferred.

Albeit some very well known *multi-objective optimization evolutionary algorithms* (MOEAs) were available, the author of this thesis wanted to compare our algorithm *SFGA* (de Toro et al., 2004) with those state-of-the-art MOEAs, namely with *SPEA2* (Zitzler et al., 2002) and with *NSGA-II* (Deb et al., 2000). After introducing some improvements to SFGA, we proposed a new MOEA called *SFGA2* (Cámara et al., 2008a). The aim of this chapter is to examine these four MOEAs, paying special attention to the new SFGA2, and to offer a comparison among them.

Hence, this chapter is structured in the following way. Firstly, in Subsection 3.1.1 the SFGA algorithm is reviewed. Then, the state-of-the-art NSGA-II and SPEA2 are briefly described in Subsections 3.1.2 and 3.1.3, respectively. Section 3.2 is devoted to the proposed SFGA2. Finally, the results of comparing these algorithms can be found in Section 3.3.

3.1 Previous Existing MOEAs

In this section, the algorithm SFGA is shown along with two state-of-the-art MOEAs, NSGA-II and SPEA2. Because the two latter algorithms are widely known among researchers, they will be briefly described, and more attention is paid to SFGA, which is reviewed in first place, because it is the basis of SFGA2.

3.1.1 SFGA (Single Front Genetic Algorithm).

Before delving into details about the SFGA algorithm, we need a definition of what is a *front*.

Definition 3.1 *A* front *or* rank *is a subset of the population where all the contained solutions are mutually non-dominated.*

In Figure 3.1 there is an example of solutions divided into the two fronts to which they belong. The fronts are found in an iterative process where the first front is the subset of the non-dominated solutions from the whole population. Then, this first front is removed from the population, and a second subset of non-dominated solutions is found. This second subset is the second front. Afterwards, this second front is removed from the subpopulation and the process is repeated in the new subpopulation until all solutions have been assigned to a front.

By sorting the population into different fronts we obtain a primitive but powerful way to divide the solutions in classes. These fronts or classes are used by some algorithms to choose which solutions should be kept and carried onto the next generation.

The *Single Front Genetic Algorithm*, SFGA, is a generational MOEA that was designed to be used inside the so-called *Parallel SFGA* (de Toro et al., 2004), an algorithm to solve multi-objective optimization problems by

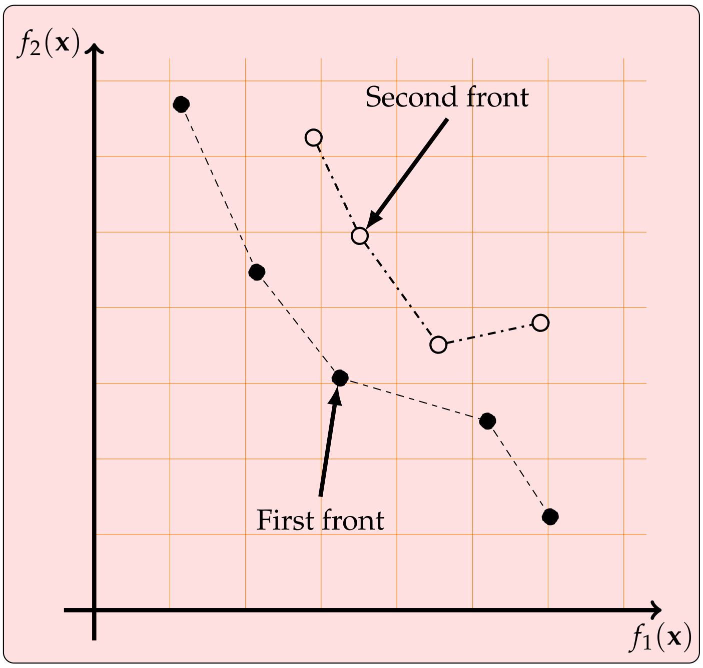

Figure 3.1: Example of fronts in a small population of solutions.

means of parallel processing. The most salient characteristic of SFGA is that it is *single-front* oriented. This means that only the non-dominated solutions belonging to the first front are taken into account to create the population for the next generation of the algorithm.

Another characteristic of SFGA is that the population size is not fixed. The algorithm has the commonly found parameter that controls the size of the population, but, in this case, it indicates the maximum number of solutions that can be kept in the population. Because only those solutions from the first front are chosen to form the new population, the new generation could have less solutions than this maximum number. In this case, the algorithm does nothing, and the next generation population has a size which is smaller than this population size parameter. On the other hand, if there are more solutions in the new generation than the allowed number, then a crowding method takes place to reduce the number of solutions within the population. An outline of SFGA is shown in Algorithm

3.1.

Algorithm 3.1: SFGA

```
Input:
    $N_{max}$ - The maximum number of solutions in the population
    $\delta_{cr}$ - The crowding distance
Output:
    $\mathbf{ND}_{out}$ - A set of non-dominated solutions
Initialize population $\mathbf{P_1}$
$i = 1$
repeat
    Mate $\mathbf{P_i}$
    Evaluate $\mathbf{P_i}$
    $\mathbf{ND}_{temp} \leftarrow$ non-dominated solutions from $\mathbf{P_i}$
    $\mathbf{ND}_{filtered} \leftarrow$ Filter $\mathbf{ND}_{temp}$ with the crowding procedure (3.2)
        and $\delta_{cr}$
    if $|\mathbf{ND}_{filtered}| > N_{max}$ then
        truncate $\mathbf{ND}_{filtered}$ to $N_{max}$
    end
    $\mathbf{P_{i+1}} \leftarrow \mathbf{ND}_{filtered}$
    $i = i + 1$
until stopping criterion is reached
$\mathbf{ND}_{out} \leftarrow \mathbf{ND}_{filtered}$
```

One of the most important parts of SFGA is the crowding mechanism used to downsize the population. This procedure is described in detail in Algorithm 3.2. The crowding method used inside SFGA, intended to maintain diversity throughout all the generations, depends on a *crowding distance* parameter, δ_{cr}.

It can be seen that once the crowding method is run on the population, the number of solutions of the resulting population N_{out} can be quite inferior to the maximum number of solutions N_{max}. The described crowding procedure shows two drawbacks:

1. It works by removing solutions that are sorted along only one of the m objective functions. This way, it removes those solutions

Algorithm 3.2: Crowding

```
Input:
    $\mathbf{P}_{in}$ - A set of, at least two, non-dominated solutions, sorted
    according to one of the $m$ objective functions: $f_1, \dots, f_m$
    $N_{in}$ - The number of solutions in $\mathbf{P}_{in}$
    $\delta_{cr}$ - The crowding distance
Output:
    $\mathbf{P}_{out}$ - The output set
    $N_{out}$ - The number of solutions in $\mathbf{P}_{out}$
1  Let $S_i$ be the $i$-th solution from sorted $\mathbf{P}_{in}$
2  begin
3      $\mathbf{P}_{out} = \{S_1, S_{N_{in}}\}$
4      $N_{out} = 2$
5      $S_{comp} = S_1$
6      for $i$=2 to $N_{in}$ do
7          if $distance(S_i, S_{comp}) \geq \delta_{cr}$ then
8              $\mathbf{P}_{out} = \mathbf{P}_{out} \cup S_i$
9              $N_{out} = N_{out} + 1$
10         end
11         $S_{comp} = S_i$
12     end
13 end
```

which are closer between themselves on a given dimension than the crowding distance. This procedure works quite well on two-dimensional problems but when the number of objective functions increases, the procedure it is not able to detect all the solutions that are close enough.

2. Due to this dependency on the sorting along one dimension the resulting output population could have solutions which should have been removed or, on the other hand, too much solutions were removed.

It is clear that due to these potential problems shown by SFGA, this algorithm had much room for improvement that could be done. Some improvements to overcome these troubles have been proposed and a new version has been developed. This version, described in Section 3.2, has been called SFGA2.

3.1.2 NSGA-II.

NSGA-II (Deb et al., 2000) is one of the best known MOEAs. The algorithm is a improved version of the former NSGA algorithm (Srinivas and Deb, 1994). NSGA-II, briefly described in this subsection, is based in the non-dominated sorting approach. This means that all the solutions in the population are sorted into the fronts to which each solution belongs. Then, the algorithm fills the next generation population with solutions from the first fronts until there is not enough space to allocate all the solutions in the current front. At that moment, a crowding procedure is used to select which solutions should be chosen according to the distances among them. This procedure is described later in this subsection. For completeness' sake, NSGA-II can be found in Algorithm 3.3.

The way in which the *fast non-dominated sorting* (Line 7) is done can be consulted in Deb et al. (2000). In order to maintain diversity, NSGA-II employs two mechanisms:

1. Density estimation. The algorithm specifies a way to calculate the distance from each solution to their two nearest solutions in each dimension. Figure 3.2 illustrates this idea. In the figure, it is shown how to calculate the distance from c to its nearest neighbours in each dimension inside its front: b and d. Because this is a two-dimensional problem the nearest neighbours are usually the same for the first and second objective, but that is not always the rule. For example, if Figure 3.2 represents all the non-dominated solutions in the first front for a three-dimensional problem projected onto the X-Y plane, then the closest neighbours to c along the X axis are b and d but along the Y axis the closest neighbours are d and h, instead of b. When the closest solutions have been found, the distances for all the dimensions are reduced to give only the crowding distance, $t_{distance}$, for this solution.

2. Crowded comparison operator. This operator guides the selection process at the various stages of the algorithm. Assuming that every

Algorithm 3.3: NSGA-II

```
Input:
    $N$ - Maximum number of solutions
Output:
    $\mathbf{ND}_{out}$ - Set of non-dominated solutions
1  Initialize randomly population $\mathbf{P}_1$ and offspring population $\mathbf{Q}_1$
2  $i = 1$
3  repeat
4      Evaluate $\mathbf{P}_i$
5      $\mathbf{Q}_i \leftarrow$ mating of $\mathbf{P}_i$
6      $\mathbf{R}_i \leftarrow \mathbf{P}_i \cup \mathbf{Q}_i$
7      $\mathcal{F} \leftarrow$ fast non-dominated sorting of $\mathbf{R}_i$
       // $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots\}$ contains the non-dominated fronts
          of $\mathbf{R}_i$
8      $\mathbf{P}_{i+1} = \emptyset$
9      $t = 1$
10     repeat
11         Assign crowding distance to $\mathcal{F}_t$
12         $\mathbf{P}_{i+1} \leftarrow \mathbf{P}_{i+1} \cup |\mathcal{F}_t|$
13         $t = t + 1$
14     until $|\mathbf{P}_{i+1}| + |\mathcal{F}_t| < N$
15     Sort $\mathcal{F}_t$ with $\prec_n$
16     $\mathbf{P}_{i+1} \leftarrow \mathbf{P}_{i+1} \cup \mathcal{F}_t[1 : (N - |\mathbf{P}_{i+1}|)]$
17     $\mathbf{Q}_{i+1} \leftarrow$ create new population from $\mathbf{P}_{i+1}$
18     $i = i + 1$
19 until stopping criterion is reached
20 $\mathbf{ND}_{out} \leftarrow \mathbf{P}_i$
```

solution t in the population has two attributes: (1) t_{rank}, the front it belongs to; and (2) $t_{distance}$, its crowding distance. A partial order relation $\prec_n$ among the solutions is defined the following way:

$a \prec_n b$ iff $(a_{rank} < b_{rank})$ OR $((a_{rank} == b_{rank})$ AND $(a_{distance} > b_{distance}))$.

This means that a solution c is better than a solution d if c belongs

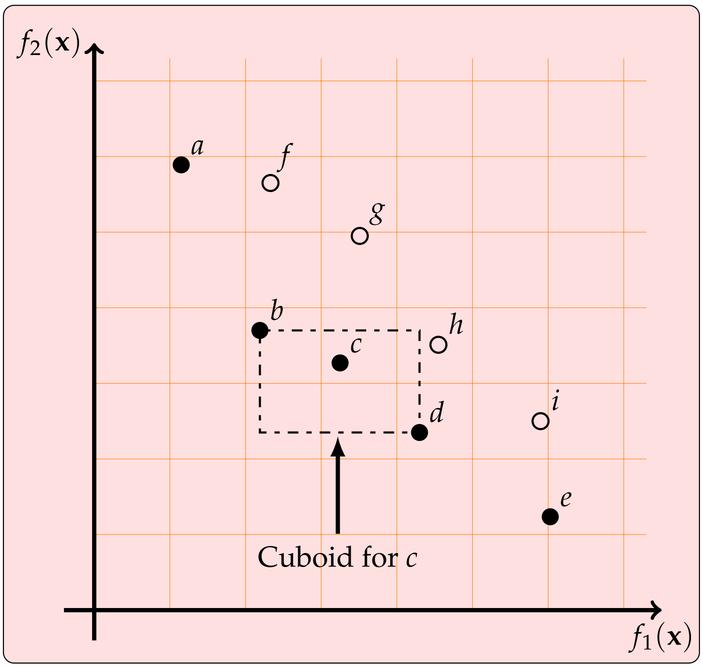

Figure 3.2: Calculation of the crowding distance for the first front.

to a better rank than d or if they belong to the same rank but the crowding distance of c is better (bigger) than that one of d.

The crowding distance of every solution is used to select the solutions that should be kept from a certain front that is too big to be copied entirely to the next generation population $\mathbf{P}_{i+1}$ (Line 16). Furthermore, the distance calculated for every solution in the next generation population $\mathbf{P}_{i+1}$ is used to guide the mating process to create the next generation offspring population $\mathbf{Q}_{i+1}$. This is so because the binary tournament selection chooses the winning solution according to the $\prec_n$ ordering between the contending solutions. Thus, those solutions with bigger crowding distance are more likely to be selected for the next offspring generation $\mathbf{Q}_{i+1}$.

3.1.3 SPEA2.

SPEA2 (Zitzler et al., 2002), the second version of the *Strength Pareto Evolutionary Algorithm*, is another MOEA very well known to researchers.

Algorithm 3.4: SPEA2

```
Input:
    $N$ - Maximum number of solutions in the population $\mathbf{P}$
    $\overline{N}$ - Maximum number of solutions in the archive $\overline{\mathbf{P}}$
Output:
    $\mathbf{ND}_{out}$ - Set of non-dominated solutions
1  Initialize randomly population $\mathbf{P}_1$
2  $\overline{\mathbf{P}}_1 \leftarrow \emptyset$
3  $i = 1$
4  repeat
5      Evaluate the solutions in $\mathbf{P}_t$ and $\overline{\mathbf{P}}_t$
6      Copy all non-dominated solutions in $\mathbf{P}_t$ and $\overline{\mathbf{P}}_t$ to $\overline{\mathbf{P}}_{t+1}$
7      if $|\overline{\mathbf{P}}_{t+1}| > N_{archive}$ then
8          Reduce $\overline{\mathbf{P}}_{t+1}$ by using the truncation operator
9      else if $|\overline{\mathbf{P}}_{t+1}| < N_{archive}$ then
10         Fill $\overline{\mathbf{P}}_{t+1}$ with dominated solutions from $\mathbf{P}_t$ and $\overline{\mathbf{P}}_t$
11     end
12     $t = t + 1$
13     if stopping criterion is met then
14         $\mathbf{ND}_{out} \leftarrow$ non-dominated solutions from $\overline{\mathbf{P}}_t$
15     else
16         Mate $\overline{\mathbf{P}}_t$
17     end
18 until stopping criterion is reached
```

SPEA2 has a complex way to assign fitness to each solution. The complexity of this process is derived from the fact that SPEA2 is not front-oriented, and so it evaluates all the solutions as possible candidates in every generation. Thus, each solution i in the archive $\overline{\mathbf{P}}_t$ and in the population $\mathbf{P}_t$ is assigned a *strength* value $S(i)$ that represents the number of solutions that it dominates. This is defined as:

$$S(i) = |\,\{j \text{ such that } j \in \mathbf{P}_t \cup \overline{\mathbf{P}}_t \wedge i \succ j\}\,| \tag{3.1}$$

where $i \succ j$ means that j is strictly dominated by i.

Additionally, a *raw fitness* $R(i)$ is defined for every solution based on the strength $S(i)$ value:

$$R(i) = \sum_{j \in \mathbf{P}_t \cup \overline{\mathbf{P}}_t \wedge i \succ j} S(j) \tag{3.2}$$

It can be easily seen that $R(i)$ is a sum of the strength values of all the solutions that dominate i. Thus, $R(i)$ is to be minimized and a non-dominated solution has a $R(i)$ value of 0. Furthermore, SPEA2 uses an estimation of the density information of the solutions. This information is used to choose between solutions that could have the same raw fitness values. The estimation of this density information is an adaptation of the k-th nearest neighbour method (Silverman, 1986), where the density of a solution is a function of the distance to the k-th nearest solution. In more detail, for each solution i the distances to all other solutions in archive and population are stored in a list. Then, the list is sorted in increasing order, and the k-th element in the list contains the distance sought and denoted as σ_i^k. It is suggested to use a value of k equal to the square root of the sample size[1], i.e. $k = \sqrt{N + \overline{N}}$ (Zitzler et al., 2002; Silverman, 1986). Therefore, the corresponding density for solution i is defined as:

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{3.3}$$

A value of 2 in the denominator is added to make sure that it is always greater than the numerator and so the resulting $D(i) < 1$. Finally, a *fitness* value $F(i)$ for a solution i is defined as:

$$F(i) = R(i) + D(i) \tag{3.4}$$

[1]Although, Zitzler et all recommend to use a value of $k = \sqrt{N + \overline{N}}$, in the implementations that they keep online (Laumanns, 2001) they use $k = 1$

Environmental Selection.

Another characteristic in SPEA2 that should be explained is the environmental selection. It takes place when non-dominated solutions are copied to the next generation archive $\overline{\mathbf{P}}_{t+1}$ in Line 6 of Algorithm 3.4. Since the archive size is fixed in every generation (i.e. there should be exactly $\overline{N}$ solutions in it) three situations are possible, as it is shown in the algorithm:

1. $\overline{\mathbf{P}}_{t+1} = \overline{N}$. The size of the new generated archive equals its maximum size, and then nothing is done.

2. $\overline{\mathbf{P}}_{t+1} < \overline{N}$. There is still space for more solutions in the archive. Then, the best non-dominated solutions in $\mathbf{P}_t$ and $\overline{\mathbf{P}}_t$ are copied to the next generation archive. This is done by choosing those $\overline{N} - |\overline{\mathbf{P}}_{t+1}|$ solutions from $\mathbf{P}_t \cup \overline{\mathbf{P}}_t$ with the smallest $F(i)$ values such that $F(i) \geq 1$ (because we already know that the solutions are dominated).

3. $\overline{\mathbf{P}}_{t+1} > \overline{N}$. In this case, $|\overline{\mathbf{P}}_{t+1}| - \overline{N}$ solutions must be removed from the archive. In order to do so, an archive truncation procedure takes place in the archive. It does so by iteratively removing one solution from $\overline{\mathbf{P}}_{t+1}$ until no more solutions have to be removed. At every iteration the truncation operator chooses for removing that solution i which has the minimum density distance σ_i^k to its k-th nearest neighbour in $\overline{\mathbf{P}}_{t+1}$. If there are more than one solution with minimum distance, then the second smallest distance, *(k-1)*-th, is checked and so forth, until the tie is broken. Mathematically, this is expressed as finding the solution i that satisfies:

$$\begin{aligned}
&\forall j \in \overline{\mathbf{P}}_{t+1} \\
&i \leq_d j :\Leftrightarrow \forall k : 0 < k < |\overline{\mathbf{P}}_{t+1}| : \sigma_i^k = \sigma_j^k \qquad \vee \\
&\qquad\qquad \exists k : 0 < k < |\overline{\mathbf{P}}_{t+1}| : \left[\left(\forall l : 0 < l < k : \sigma_i^l = \sigma_j^l\right) \wedge \sigma_i^k < \sigma_j^k\right].
\end{aligned}$$

This truncation procedure is run only when there are more solutions in the archive $\overline{\mathbf{P}}_t$ than its maximum size $\overline{N}$.

3.2 SFGA2

Drawing from the experience acquired with NSGA-II (Deb et al., 2000) and, especially, with SPEA2 (Zitzler et al., 2002), SFGA has been adapted to use fitness values to guide the selection of solutions from the current population that should be carried onto the next generation. SFGA explicitly includes only non-dominated solutions in the next generation population and so the modifications introduced to it should adhere to that principle. Thus, the fitness value for any solution cannot be based on the rank of that solution or the number of solutions it dominates but only on a measure of the solutions lying near to it, or in other words, the crowdedness of its neighbourhood.

In this way, a fitness value has been defined using a crowding distance already present in the SFGA algorithm. The fitness, λ_i, of a solution S_i, is the average distance to that solution from all other solution which lie inside that crowding distance, δ_{cr}, but with a little modification. In order to give more importance to solutions which lie in less crowded areas, the number of surrounding solutions is included in the denominator of the expression. Doing so, the average distance to other solutions is further divided by the number of such solutions which makes that value to decrease when the number of surrounding solutions increases. Hence, λ_i is given by:

$$\lambda_i = \frac{\sum\limits_{S_k \in \mathbf{C}_i} \text{distance}(S_i,\, S_k)}{|\mathbf{C}_i|^2} \tag{3.5}$$

where $\text{distance}(S_i, S_j)$ is the Euclidean distance from solution i to solution j and $\mathbf{C}_i = \{S_j \colon \text{distance}(S_i, S_j) < \delta_{cr} \quad \forall j\, 1 \leq j \leq n \quad \wedge \quad i \neq j\}$.

However, after using this proposed fitness value on some test data we obtained unexpected and undesirable sets of non-dominated solutions where diversity was not maintained.

The reason that the set of selected solutions with the latter approach does not maintain the diversity is that only following the crowding criterion is not enough. We have introduced some randomness in this procedure in order to avoid this situation of getting stuck with solutions which are not showing any diversity. Randomness has been added to the fitness value by reducing the fitness value of every m-th solution, being m the number of objectives. This reduced fitness value is calculated with Eq.

(3.5) by multiplying it by a random value between α and $(1-\alpha)$, with $\alpha < 0.5$.

The Diversity Maintenance Procedure Outlined.

Contrary to SFGA, in SFGA2 the population size is kept constant among generations. In order to choose which solutions should be removed or added to the next generation population the new defined fitness value (3.5) will be used. The diversity procedure is described in Algorithm 3.5.

This procedure can be greatly optimized by combining together some steps. Nevertheless, in this thesis they are shown separately in order to be understandable.

Differences with SPEA2. As it was said earlier, this procedure bears a strong resemblance to the procedure implemented in SPEA2, but there are a few differences. First, this procedure calculates a fitness value only from the distances of the neighbour solutions. Second, these neighbour solutions are chosen based solely on the crowding distance whilst in SPEA2 a fixed set of the nearest K solutions is used.

3.2.1 The Evolutionary Algorithm SFGA2.

SFGA had to be changed accordingly to take into account the new procedure for maintaining the diversity. In its new version, the size of the population (N) is kept constant among generations. At the end of each generation, from this population, only the best N solutions according to the fitness value, will be selected to survive to the next generation. As the original SFGA, only the non-dominated solutions are taken into account to generate the offspring or to be taken onto the next generation. The new SFGA2 can be found in Algorithm 3.6:

However, the description of SFGA has required more space than the descriptions of NSGA-II and SPEA2, SFGA-2 runs much faster then they do, as it will be seen in the next section.

Algorithm 3.5: Procedure for maintaining diversity

```
Input:
    C - Set of input solutions
    $N$ - Number of input solutions in C
    $N_{out}$ - Number of output solutions
    $\delta_{cr}$ - Crowding distance
    $\alpha$ - The random factor
Output:
    $\mathbf{C}_{out}$ - Set of output solutions
1  Let $m$ be the number of objective functions to optimize.
2  begin
3      Build $\mathbf{Z}_{NxN} : \mathbf{Z}_{i,j} = \mathbf{Z}_{j,i} = \text{distance}(S_i, S_j)$
4      foreach solution $S_i \in \mathbf{C}$ do
5          $\mathbf{H}(S_i) = \left\{S_j : distance(S_i, S_j) < \delta_{cr} \; \forall j, 1 \leq j \leq N \wedge j \neq i\right\}$
6          $\overline{D}(S_i) = \frac{\sum_{S_j \in H(S_i)} \text{distance}(S_i, S_j)}{|H(S_i)|}$
7          $\lambda_i = \frac{\overline{D}(S_i)}{|H(S_i)|}$
8          if $i \mod m = 0$ then
9              $\lambda_i = \lambda_i * \text{random}(\alpha, 1 - \alpha)$
10         end
11     end
12     Sort C in descending order according to fitness values.
13     $\mathbf{C}_{out} \leftarrow N_{out}$ first solutions of C
14     Select the $N_{out}$ first solutions as the output set.
15 end
```

3.3 Results

In order to compare the performance of the new algorithm along with the other MOEAs, two multi-objective functions will be used: the first test function has two objective functions whereas the second one has three objectives to optimize. As we are interested only in the multi-objective part of the optimization process, the test functions are not required to show any dynamical behaviour. Thus, the chosen test functions are just

Algorithm 3.6: SFGA2

```
Input:
    $\mathbf{C}^N_{in}$ - Input population
    $N$    - Number of solutions in the population $\mathbf{P}$
Output:
    $\mathbf{C}^N_{out}$ - Output population
begin
    $\mathbf{P}^N_{temp} \leftarrow \mathbf{C}^N_{in}$
    while stop criterion is not reached do
        $\mathbf{P}^N_{offspring} \leftarrow \mathbf{P}^N_{temp}$ after applying the crossover and
          mutation operators
        Evaluate $\mathbf{P}^N_{temp}$ and $\mathbf{P}^N_{offspring}$
        $|\mathbf{ND}^{2N}_t| \leftarrow$ non-dominated$\{\mathbf{P}^N_{temp} \cup \mathbf{P}^N_{offspring}\}$
        if $|\mathbf{ND}^{2N}_t| > N$ then
            foreach $s \in \mathbf{ND}^{2N}_t$ do
                Calculate fitness for $s$ with (Eq. 3.5)
            end
            $\mathbf{P}^N_{temp} \leftarrow$ best $N$ solutions according to $\lambda_i$
        else
            $\mathbf{P}^N_{temp} \leftarrow \mathbf{ND}^{2N}_t$
            Fill $\mathbf{P}^N_{temp}$ with $N - |\mathbf{ND}^{2N}_t|$ random solutions from
            $\mathbf{P}^N_{temp} \cup \mathbf{P}^N_{offspring}$
        end
    end
    $\mathbf{C}^N_{out} \leftarrow \mathbf{P}^N_{temp}$
end
```

stationary versions of the dynamic functions FDA1 and FDA4 that can be seen in further detail in Section 2.5.

The parameters used for all the MOEAs are compiled in Table 3.1. The measures that will be used here are those which treat mainly with the multi-objective nature of optimization algorithms. Namely, *hypervolume* indicator, *spacing*, *IGD* and *closeness* will be used. They are explained in

Table 3.1: Parameters used in the tests

	FDA1	FDA4
Population size	100	500
Number of generations	500	1000
Number of independent runs	30	30
Mutation rate	0.015	0.002
Crossover rate	0.8	0.8
Crowding distance	0.015	0.075

more detail in what follows:

- The *hypervolume* indicator (Zitzler et al., 2000) is a measure which gives the area covered by a given approximation set from a reference point.
- *Spacing* (Coello et al., 2007) gives a measure of how well the solutions from the approximation set are spaced.
- *IGD, Inverted Generational Distance*, gives the distance to every solution from a subset of the real Pareto front to the nearest solution in the approximation set. *IGD* is used only for the two-dimensional test function.

For the three-dimensional test function instead of using *IGD*, a more straightforward measure will be used. Due to the fact that the FDA4 function corresponds to one octave of the sphere of radius one with center in the origin, there is a simple way to calculate how close are the solutions from the approximation set to the real Pareto front. The real Pareto front is comprised of all points whose Euclidean distance to the origin equals one. Thus, *closeness* is obtained in the following way:

$$Closeness = \sum_{i=1}^{|\mathbf{A}|} \{||S_i|| - 1\} \tag{3.6}$$

where $\mathbf{A}$ is the approximation set and S_i is the *i-th* solution from $\mathbf{A}$ and $||S_i||$ is the norm of S_i.

To have a significant level of statistical confidence on the obtained results, the algorithms are run thirty times, and each time the population is randomly created and a new random seed is set. Once the algorithms have been run, the afore-mentioned performance measures are calculated for each approximation set. For every performance measure a boxplot is used to summarize the statistical indicators for every algorithm. Then, the analysis of variance (ANOVA) procedure is run on the data for the four algorithms in order to check if it is possible to reject the null hypothesis, which means that the data from the four groups come from the same source. If the ANOVA *p-value* on the four data groups allows us to reject the null hypothesis with a certain confidence level (smaller than $0,05$) then we have to differ which groups show also statistical significant difference but compared in pairs.

In order to do these pairwise comparisons, the Tukey's HSD (Honestly significant difference) post-hoc test (Scheffé, 1999) is the most commonly used test. HSD test is preferred over other pairwise statistics (such as Bonferroni tests) because it produces less type I errors (Scheffé, 1999). If the ANOVA is negative (the *p-value* is above our desired confidence level of 0,05), then further inference on the data will not be possible. ANOVA can be used because after thirty independent runs the results gathered from the algorithms are supposed to follow a normal distribution (Demšar, 2006). If the results distribution would not be normal, then other kind of statistics should have to be used instead of ANOVA. Detailed discussion for those cases is offered in Demšar (2006) and García and Herrera (2008).

3.3.1 Stationary FDA1.

Firstly, the algorithms are tested on a two-dimensional problem which is the stationary version of the FDA1 problem. The first performance measure used in the results for FDA1 is the hypervolume indicator (Zitzler et al., 2000). Figure 3.3 shows the boxplots for the four different algorithms.

ANOVA tells us that the obtained results for hypervolume are significant enough to discard the null hypothesis that all the data come from the same source.

Then, six pairwise Tukey's HSD post-hoc tests are carried out on the

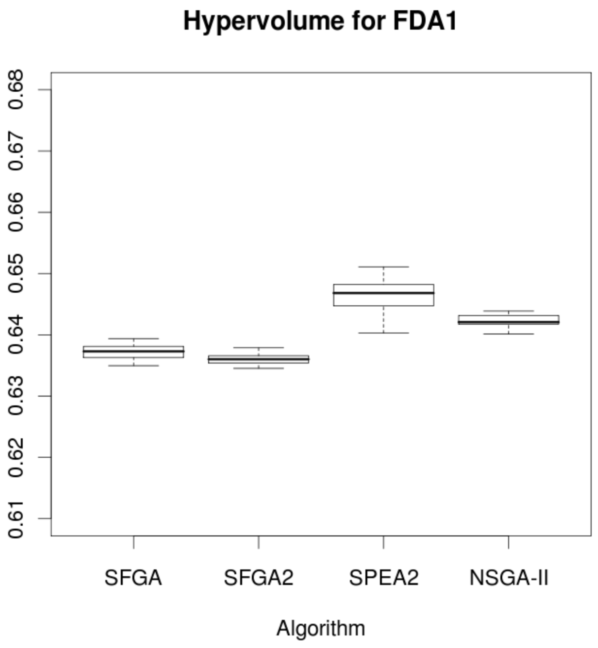

Figure 3.3: Hypervolume for stationary FDA1 problem. Bigger is better.

data. The results from these tests are compiled in Table 3.2. It can be seen that all the results are significant among themselves.

Therefore, conclusions from the observed results can be inferred for all the pairwise combinations of the four algorithms. Hence, we can say that SPEA2 is the algorithm that performed best for stationary FDA1. The other algorithms followed SPEA2 closely, and the difference between them is quite small. Namely, SPEA2 was better than SFGA2 by only 2%.

Next, the *spacing* measure is employed to know how well spaced the distributions of the solutions are. A summary of the main statistical measures for these data is shown in the boxplots in Figure 3.4. Again, ANOVA is run over the four data groups, and it allows us to reject the null hypothesis with $p << 0.001$.

In this case, the post-hoc tests (Table 3.3) found significant differences between all the possible pair combinations except for the pair between

Table 3.2: HSD post-hoc tests for *hypervolume* for FDA1

Pair compared	Difference	$HSD_{0.05}$	Significant
SFGA - SFGA2	0.00122540	0.00109259	YES
SFGA - SPEA2	0.00914753	0.00109259	YES
SFGA - NSGA-II	0.00495563	0.00109259	YES
SFGA2 - SPEA2	0.01037290	0.00109259	YES
SFGA2 - NSGA-II	0.00618103	0.00109259	YES
SPEA2 - NSGA-II	0.00419190	0.00109259	YES

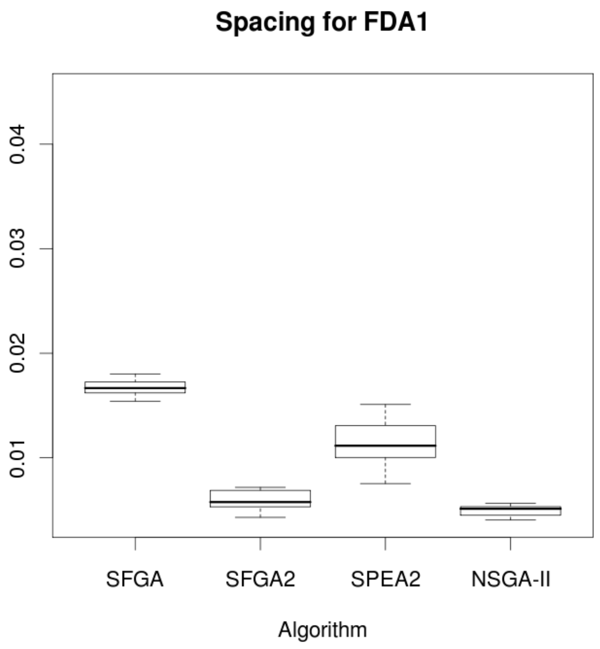

Figure 3.4: Spacing measure for stationary FDA1 problem. Bigger is better.

SFGA2-SPEA2. On the light of these Tukey's HSD results, we could not state that SPEA2 did better than SFGA2 did and further study should have to be done with other statistics or more runs should be added to our study. But even with the result for that pairwise comparison, we can conclude that SFGA performed better than the other three algorithms and NSGA-II was the algorithm that did worst. Again, the differences between the algorithms are very small and other factors should be taken into account before choosing which algorithm to use.

Table 3.3: HSD post-hoc tests for *spacing* for FDA1

Pair compared	Difference	$HSD_{0.05}$	Significant
SFGA - SFGA2	0.00736523	0.00325957	YES
SFGA - SPEA2	0.00504176	0.00325957	YES
SFGA - NSGA-II	0.01178540	0.00325957	YES
SFGA2 - SPEA2	0.00232347	0.00325957	NO
SFGA2 - NSGA-II	0.00442014	0.00325957	YES
SPEA2 - NSGA-II	0.00674361	0.00325957	YES

The last performance measure used with FDA1 is *IGD*. The obtained results are summarized in the boxplots shown in Figure 3.5.

The result of ANOVA indicates that there is enough significance to reject the null hypothesis. Because the null hypothesis can been rejected, the Tukey's HSD post-hoc tests are run for the IGD results and the HSD outcomes are shown in Table 3.4.

Table 3.4: HSD post-hoc tests for *IGD* for FDA1

Pair compared	Difference	$HSD_{0.05}$	Significant
SFGA - SFGA2	0.000896259	0.000835398	YES
SFGA - SPEA2	0.003289890	0.000835398	YES
SFGA - NSGA-II	0.002767660	0.000835398	YES
SFGA2 - SPEA2	0.004186150	0.000835398	YES
SFGA2 - NSGA-II	0.003663920	0.000835398	YES
SPEA2 - NSGA-II	0.000522230	0.000835398	NO

It can be seen that the pair SPEA2 and NSGA-II is not comparable and no conclusion can be drawn between themselves. In spite of that we can

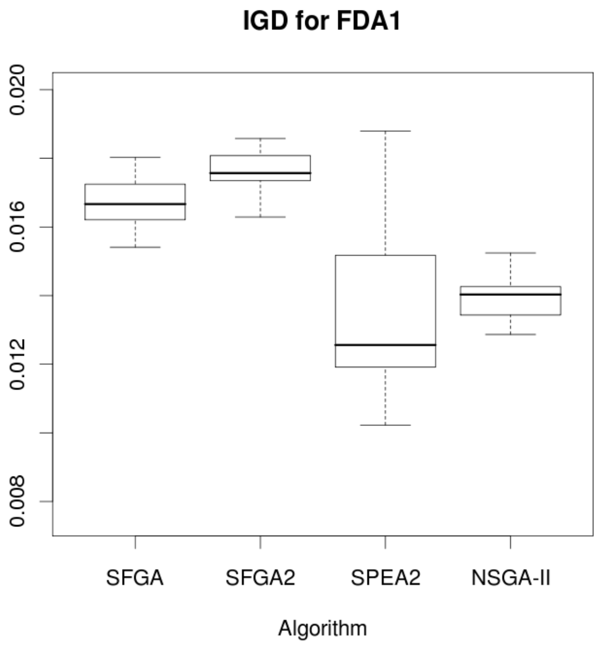

Figure 3.5: IGD measure for stationary FDA1 problem. Smaller is better.

state that SPEA2 and NSGA-II behaved better than SFGA and SFGA2. Also, it can be said that although SPEA2 did better than SFGA and SFGA2 even though the boxplots show that the SPEA2 results suffered from a high standard deviation. Again, the results did not differ greatly from one algorithm to the other ones. These results for *hypervolume*, *spacing* and *IGD* will be evaluated as a whole along with the time employed by each algorithm at the end of this chapter, after seeing some performance measures on the stationary FDA4 problem.

3.3.2 Stationary FDA4.

In this case, the test function is the three-dimensional function FDA4 fixed to a stationary state where $\tau = 0$ for all the generations and all runs.

The *hypervolume* indicator gave the results summarized in the box-plots shown in Figure 3.6.

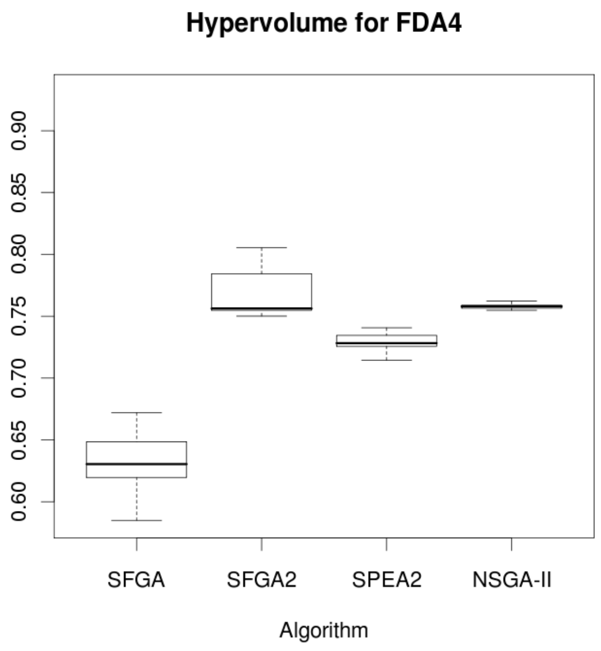

Figure 3.6: Hypervolume for stationary FDA4 problem. Bigger is better.

Once more, ANOVA allows us to reject the null hypothesis. Therefore, the pairwise post-hoc tests are calculated for the *hypervolume* indicator and their corresponding results are shown in Table 3.5.

These results tell us that with a confidence level of $p << 0.05$ we can infer conclusions for all the pairwise comparisons from the data groups available to us. Hence, we can conclude that SFGA2 was the algorithm that performed best of all. In addition, SPEA2 and NSGA-II also performed well and their results are located close to the SFGA2 results. On the other hand, SFGA did poorly on FDA4 with regard to the *hypervolume* indicator.

The following measure used with FDA4 is *closeness*. In this case, *closeness* is calculated in a very straightforward way because the Pareto front

Table 3.5: HSD post-hoc tests for *hypervolume* for FDA4

Pair compared	Difference	$HSD_{0.05}$	Significant
SFGA - SFGA2	0.1476520	0.0159358	YES
SFGA - SPEA2	0.0978277	0.0159358	YES
SFGA - NSGA-II	0.1263000	0.0159358	YES
SFGA2 - SPEA2	0.0498246	0.0159358	YES
SFGA2 - NSGA-II	0.0213525	0.0159358	YES
SPEA2 - NSGA-II	0.0284721	0.0159358	YES

is one octave of the sphere with radius one. The *closeness* values obtained for the different runs of each algorithm are summarized in the statistical indicators shown in the boxplots from Figure 3.7.

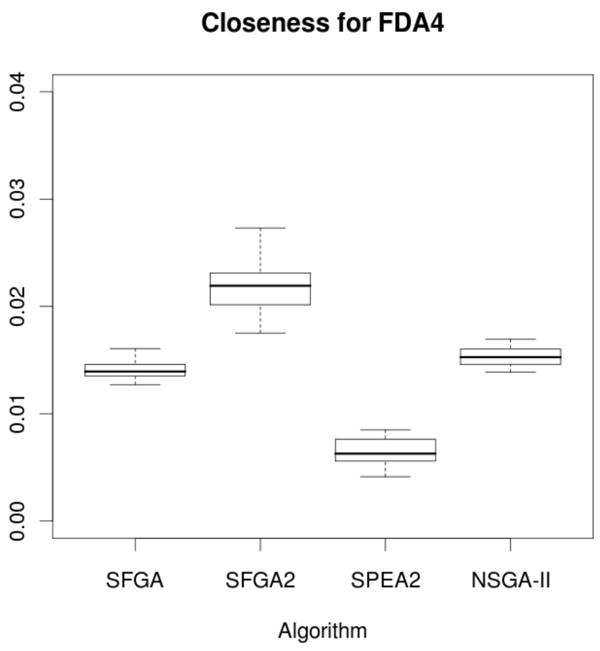

Figure 3.7: *Closeness* for stationary FDA4 problem. Smaller is better.

After applying ANOVA the null hypothesis that all the means for the *closeness* performance measure are equal can be rejected. Now, it is turn for the Tukey's HSD post-hoc tests for pairwise comparisons among all the algorithms. The results of these post-hoc tests are gathered in Table 3.6.

Table 3.6: HSD post-hoc tests for *closeness* for FDA4

Pair compared	Difference	$HSD_{0.05}$	Significant
SFGA - SFGA2	0.00375564	0.0019481	YES
SFGA - SPEA2	0.00198469	0.0019481	YES
SFGA - NSGA-II	0.00886471	0.0019481	YES
SFGA2 - SPEA2	0.00574033	0.0019481	YES
SFGA2 - NSGA-II	0.00510906	0.0019481	YES
SPEA2 - NSGA-II	0.01084940	0.0019481	YES

From the results contained in Table 3.6 we can freely infer conclusions for any pair of algorithms, because all passed the Tukey's HSD tests. Consequently, it can be said that SPEA2 was the algorithm that performed best among all the analysed algorithms. In addition, SPEA2 was a clear winner with respect to the *closeness* performance measure for FDA4.

The last performance measure used with the stationary FDA4 function is *spacing*. For *spacing* the statistical summaries are provided by the boxplots from Figure 3.8.

Once more ANOVA rejects the null hypothesis and Tukey's HSD post-hoc tests are carried out on the data. The results of these post-hoc tests can be found in Table 3.7.

In this case an interesting situation arises. Firstly, we can conclude that, according to the *spacing* measure, SFGA was the best algorithm with a clear distance to the other three competing algorithms. But this is only according to the *spacing* measure. The reason behind this result is that SFGA did so bad with the three-dimensional problem, as seen in the *hypervolume* indicator in Figure 3.6, that the solutions provided by SFGA were so far from the real Pareto front that they are also far among themselves and so the *spacing* measure gives a higher value.

Then we can turn our attention to the other three algorithms. From

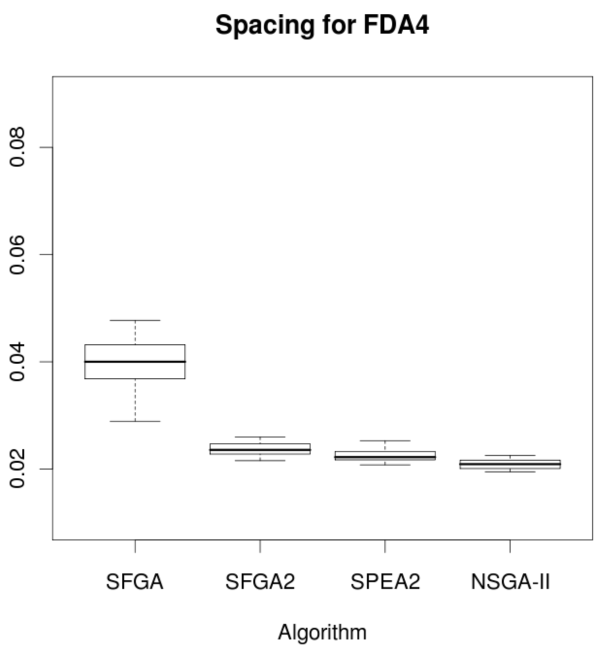

Figure 3.8: *Spacing* for stationary FDA4 problem. Bigger is better.

Table 3.7: HSD post-hoc tests for *spacing* for FDA4

Pair compared	Difference	$HSD_{0.05}$	Significant
SFGA - SFGA2	0.01581230	0.00206973	YES
SFGA - SPEA2	0.01729440	0.00206973	YES
SFGA - NSGA-II	0.01893350	0.00206973	YES
SFGA2 - SPEA2	0.00148203	0.00206973	NO
SFGA2 - NSGA-II	0.00312120	0.00206973	YES
SPEA2 - NSGA-II	0.00163916	0.00206973	NO

Table 3.7 we know that conclusions cannot be extracted for the pair-wise comparisons between SFGA2 and SPEA2, and between SPEA2 and NSGA-II, respectively. Fortunately, the pair SFGA2 - NSGA-II has passed

the HSD post-hoc test. Hence, a conclusion can be made stating that SFGA2 performed better than NSGA-II with respect to the *spacing* performance measure.

Run-time of the algorithms

On the light of the results shown above it could be said that the best performing algorithm for FDA1 is SPEA2. But we should take into account one more factor before stating a firm conclusion.

Due to the fact that the algorithms are to be used to optimize dynamic problems, it seems reasonable to look at how much time was employed by each algorithm to complete the same task, namely to optimize the given problem with a fixed population and a fixed number of iterations for thirty independent runs. Figure 3.9 shows the amount of time, in minutes, that each algorithm employed for the afore-mentioned assigned task.

On the contrary, it could not be said that one of the shown algorithms is better than the others according to most of the performance measures used. While SFGA2, was the best for *spacing*[2], it was SPEA2 the algorithm that got the best closeness values. But for *hypervolume* the best results were obtained by SFGA2 and NSGA-II. At this moment, we proceed to take into account the time, shown in Figure 3.10, that each algorithm needed to complete the same task.

From Figure 3.9 it can be seen that the runtime shown by SFGA and SFGA2 is certainly smaller than the time that SPEA2 or NSGA-II needed to complete the same task. This difference in runtime is even more clear in Figure 3.10 where NSGA-II has a much longer runtime than SPEA2. At the same time, SPEA2 is considerably slower than SFGA2.

3.3.3 Conclusion from the Results.

From the results that have been offered in the previous subsection it is possible to claim that SFGA2 is a better algorithm than SFGA for problems with more than two objective functions. This has been shown by the

[2]It is has been noted already that SFGA was the best for *spacing* but that this result was not relevant due to problems that SFGA showed with the other measures.

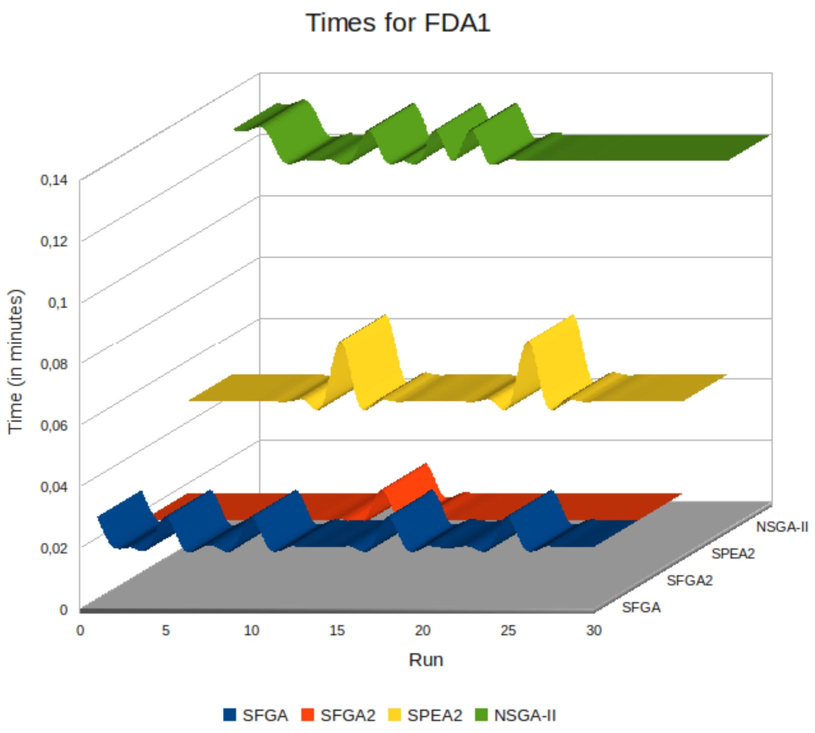

Figure 3.9: Times for stationary FDA1 problem.

results for the stationary FDA4 problem. In addition, SFGA2 did not perform worse than SFGA for the two objective problem. Hence, we have succeed in designing an improved version of SFGA to deal with three or more objective problems while not worsening the results for two dimensional problems.

Furthermore, taking into account our goal to use these algorithms in a dynamic optimization scenario SFGA2 has proved itself a decent competitor for the state-of-the art algorithms SPEA2 and NSGA-II. The reason behind this is that although SPEA2 and NSGA-II together got slightly better performance results than SFGA2, the latter did not so badly in the

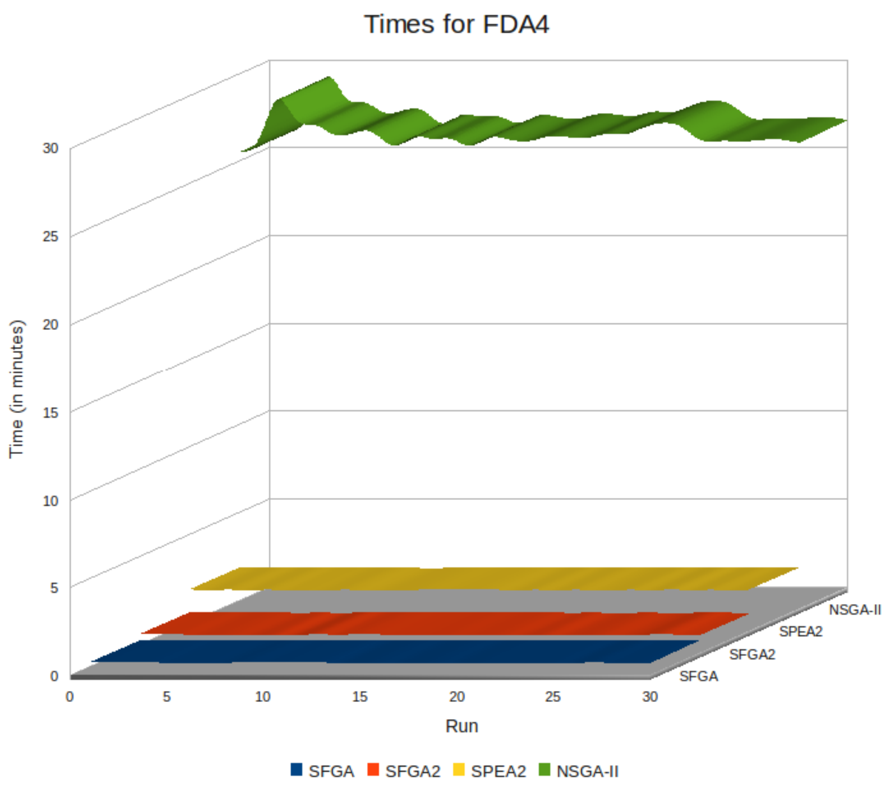

Figure 3.10: Times for stationary FDA4 problem.

performance measures. On the other hand, SFGA2 employed much less time than SPEA2 and NSGA-II did to complete the runs. This means that in dynamic environments where the algorithms usually need to deal with tight time restrictions, SFGA2 is a better option because it could produce quite good solutions while it would be expected to fulfil the most restrictive time limits.

3.4 Summary

In this chapter, a new improved version of the SFGA algorithm has been proposed. This new version, called SFGA2, has been improved to deal better with problems with more than two objective functions. Both algorithms, SFGA and SFGA2, have been tested for two test cases along with two state-of-the-art multi-objective optimization algorithms: SPEA2 and NSGA-II. The results from these tests have shown that even though SFGA2 was not the best performing algorithm, it showed a quite good behaviour with regard to the performance measures used while completing all the tests in much less time that the two state-of-the-art algorithms.

Due to the importance that time plays in dynamic optimization problems, we believe that SFGA2 is a serious alternative to SPEA2 and NSGA-II for those kind of problems.

Contributions

The improvements to SFGA that led to the development of the new SFGA2 and that have been shown throughout this chapter have been already published in the literature (Cámara et al., 2008a, 2007a).

Next Chapter

The next chapter is dedicated to analyse the different ways of parallelizing the processing of MOEAs. Moreover, two procedures are proposed. Afterwards, these two procedures are compared side-by-side.

"Learning without thinking is useless; thinking without learning is dangerous."

Confucius

4 Parallel Procedures for Dynamic Multi-objective Optimization

IN this chapter two parallel processing approaches for MOEA are introduced and reviewed. The first one is called *pdMOEA* and it is a hybrid procedure where some processes behave like workers but others behave as master processes. The second approach is a fully distributed procedure where all the optimization is carried by independent workers. In our work, data decomposition has been applied as we consider this alternative more attractive.

An introduction to parallel procedures used for solving multi-objective optimization problems by means of evolutionary algorithms can be found in Section 4.1. Afterwards, some theory to analyze the speedup provided by multi-objective parallel procedures is given in Section 4.2. In Section 4.3, the details about the pdMOEA procedure are provided. Then, the fully distributed method, pdMOEA+, is outlined in Section 4.4. Finally, results about the features and performance of these two procedures can be found in Section 4.5.

4.1 Approaches to Parallelism for MOEAs

Parallel processing can be useful to efficiently solve dynamic optimization problems with evolutionary algorithms (van Veldhuizen et al., 2003; Luna et al., 2006), not only by improving the quality of the solutions found but also by speeding up the execution times. Here, we explore the benefits of parallel processing in multi-objective dynamic problems.

Two decomposition alternatives are usually implemented in parallel evolutionary algorithms: functional decomposition and data decomposition.

The functional decomposition techniques identify tasks that may be run separately in a concurrent way. The data decomposition techniques divide the sequential algorithm in different tasks that are run on different data (i.e. the individuals of the population). Moreover, hybrids methods are also possible.

In an evolutionary algorithm, the evaluation of the objective function and the application of operators to the individuals of the population can be independently done for each individual. This allows data parallelization without modifying the convergence behavior of the sequential algorithm. The fitness evaluation for each individual in the population is usually the part with the highest computational cost. This is mainly true in non-trivial optimization problems, with big sized populations and/or individuals codified with complex data structures that require big computation times.

As a consequence, the most usual parallelization scheme is to evaluate concurrently the individuals, usually with a master-worker implementation in which every worker process evaluates a different and unique group of individuals, returning the fitness values to the master process which continues with the rest of the algorithm steps.

It is useful to review the different types of parallelization available for MOEAs (Alba, 1999; Tomassini, 1999; Cantu-Paz, 2000).

- *Master-worker*. As it has been said, in this approach there is a master process which is in charge of performing some or all the operations on the whole population. In addition, the worker processes carry some or all operations but only on a fraction of the population. In Figure 4.1 this scheme is outlined.

- *Island*. This is a coarse-grained distributed model, where all the processes carry all the operations but only on some part of the whole population. In this model, the processes can interchange some of their solutions with the neighbour processes. Figure 4.2 depicts the island approach.
- *Cellular* or *diffusion*. This is a fine-grained distributed model, in which every process hold only one solution of the population. Again, it is a common practice that cellular processes swap their solution with the solution in one of their neighbours.

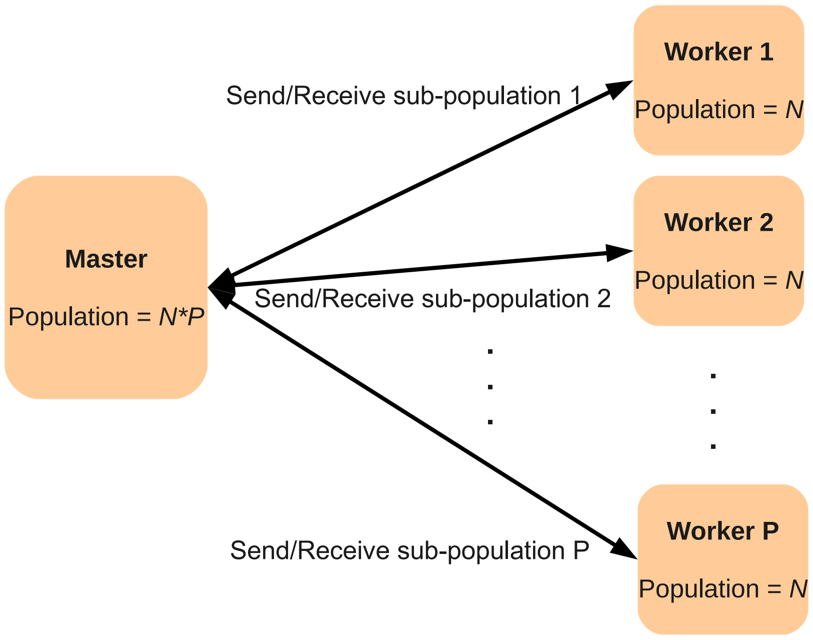

Figure 4.1: Scheme of master-worker processing for MOEAs.

In addition to these models, it is possible to develop algorithms that

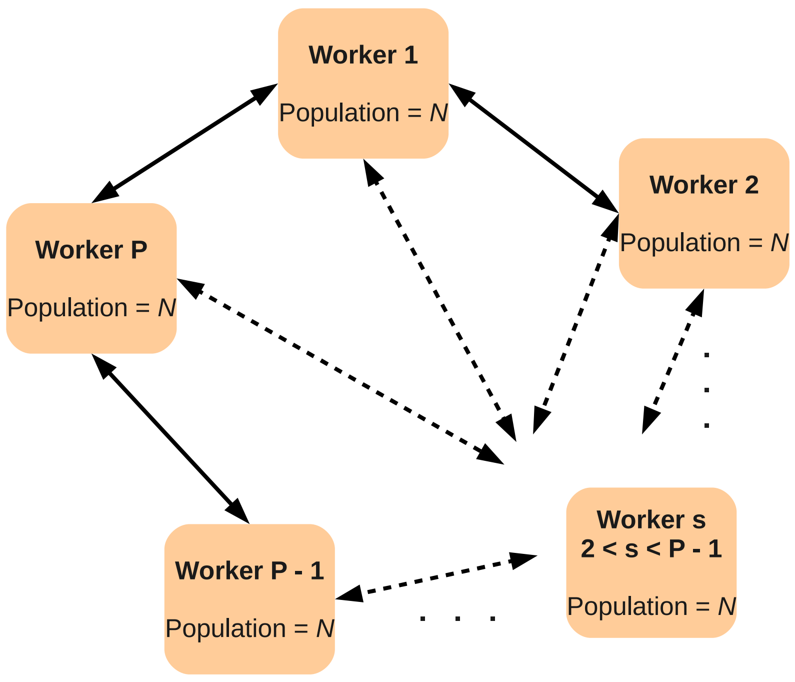

Figure 4.2: Scheme of island processing for MOEAs.

show a hybrid behaviour, where the parallelization changes between master-worker and island during the execution of the program.

Returning back to the master-worker model, if the individuals are distributed in a balanced way there could be linear speedups, but unless the evaluation of the solutions require a high computation time, the costs associated with the distribution of the data structures between the processors and the communication of the results may considerably decrease the efficiency of this kind of parallel procedures.

The selection of individuals and the diversity maintenance operations require comparisons that imply the whole population or a big part of it. This means that data parallelization at this level, specially in the case where there is not any mechanism to share information among the processes about the fitness of the individuals, modifies the behaviour of the algorithm with regard to the sequential version. Usually, it is difficult to predict the behaviour of this kind of parallelization and must be evaluated for each particular implementation. The initial population is divided into sub-populations associated to different search spaces which are evolved separately. Sometimes, individuals can be exchanged between the sub-populations (migration). This kind of parallelization could improve the diversity of the population during the algorithm convergence and lead to algorithms with better performance than the sequential versions. So, together with the advantages due to the bigger availability of memory and CPU, the evidences of bigger efficiency and diversity in the population justify the use of parallelism in the field of evolutionary algorithms.

Moreover, in multi-objective optimization the different objectives and the Pareto dominance relations have to be evaluated (de Toro et al., 2004; van Veldhuizen et al., 2003). As it has been said, the calculation of the Pareto dominance relations requires, most of the time, statistics of the whole population. Besides, the computational bottleneck in most of the applications is the evaluation of the set of objective functions, which may be parallelized by means of distributing the objective functions among processors, or with a hybrid approach in which each processor evaluates a subset of functions for a subset of the population.

After the evaluation of the objective functions, the algorithms with Pareto front-based selection usually calculate dominance and the corresponding distances as part of the mechanism for keeping diversity. This mechanism is implemented in each case, as a previous step to assign fitness values to each individual and to select the parents. The parallelization of these tasks is not easy. For example, problems appear in algorithms that usually work with small populations, PAES (Knowles and Corne, 1999), in algorithms where the calculation of distances must be done sequentially after the determination of dominance relations, PSFGA (de Toro et al., 2004), or in those algorithms where the calculation of dominance relations and distances, and the selection take place at the same time, NPGA (Horn and Nafpliotis, 1993).

4.2 A Model for Speedup Analysis

Roughly speaking, dynamic multi-objective optimization can take advantage of parallel processing in a similar way that stationary multi-objective optimization does. This includes the possibility of speeding up the capacity of the algorithm reaction. This, parallel processing reduces the time required to provide a set of non-dominated solutions near to the Pareto front earlier. Thus, dynamic optimization could also cope with problems with faster change rates.

In what follows, a model is proposed to understand these benefits. The time that a sequential evolutionary algorithm for multi-objective optimization requires is:

$$T_s = gen \times ((AMt_0) + (BM^r t_1))$$

and the time for a given parallel version executed on P processors is given by:

$$\begin{aligned} T_p = {} & genser \times ((AMt_0) + (BM^r t_1)) + \\ & genpar \times ((A(M/P)t_0) + (B(M/P)^r t_1)) + O(M,P) \end{aligned}$$

In these expressions, M is the number of individuals in the population, t_0 is the time required by the genetic operators (crossover, mutation, etc.), and t_1 is the time required by the multi-objective algorithm to determine the Pareto front and maintain an adequate distribution of individuals across it. The complexity of these operations is taken into account through the parameter r.

The parameters A and B determine the relative weight of terms depending on M and M^r, respectively. In T_s, parameter gen is the number of generations executed by the sequential algorithm. The parameters $genser$ and $genpar$ in T_p correspond, respectively, to the number of generations executed in a master processor and in each of the worker processors where the population has been divided into (M/P) individuals. If $genser = 0$, an island model is used to parallelize the algorithm, while if $genser > 1$, we have a master-worker procedure. For example, we can set different values for $genser$ and $genpar$ in our parallel procedure to implement an island model that allows the communication among the sub-populations through a master. The term $O(M,P)$ corresponds to the communication cost: it depends on the amount of individuals that processors exchange (a function of M) and on the number (and communication topology) of processors that have to communicate themselves (a function of P).

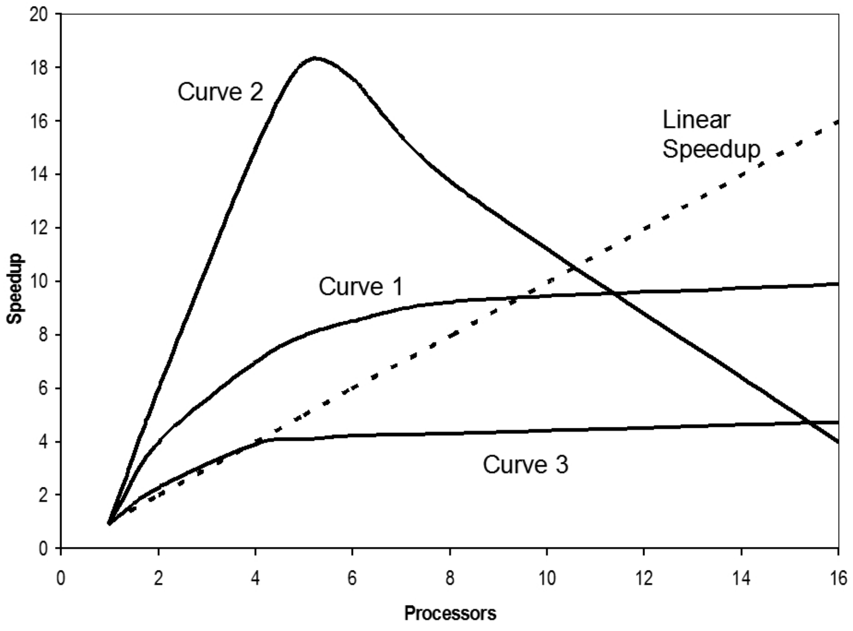

Figure 4.3: Different speedups behaviours.

This simple model allows us to explain different speedup ($S = T_s/T_p$) behaviours (Alba, 2002). Thus, if $genser + genpar < gen$ it is possible to observe super-linear speedups (as in curves 1 and 2 of Figure 4.3). This situation could appear whenever the parallel evolutionary algorithm provides, for example, better diversity conditions than the sequential implementation and a lower number of iterations is required by the parallel algorithm to get a solution with similar quality that the one obtained by the sequential algorithm.

Moreover, the effect of the communication cost can be also taken into account. Thus, in Figure 4.3, as the number of processors P increases, the speedup is lower in curve 2 than in curve 1, the communication cost is higher for curve 2, although in this curve $genpar$ is higher and $genser + genpar$ is lower than in curve 1. In Figure 4.3, curve 3 corresponds to a case where $genser + genpar > gen$.

In Alba (2002), the following taxonomy is proposed for speedup measurements in evolutionary algorithms:

- Class I. *Strong* speedup.
- Class II. *Weak* speedup:
    - A. Speedup with solution-stop.
        - (a) Versus panmixia.
        - (b) Orthodox.
    - B. Speedup with predefined effort.

This taxonomy distinguishes between the Class I or strong speedup measurements, which compare the execution times of the parallel evolutionary algorithms and the better known sequential algorithm for the problem at hand; and the Class II measurements that compare the parallel algorithm with its own sequential version executed in only one processor. Inside the Class II measurements, it is also possible to distinguish between other two types of measurements according to the way the algorithm finishes. The group A includes the measurements obtained if the algorithms finish when solutions of similar qualities are found by both, the parallel and sequential algorithms.

Whenever the measures are obtained by setting a similar number of iterations for the sequential and the parallel algorithms we have the group B of measurements. As it can be seen, in the performance model described in this section the speedup measurement considered belong to the class II and group B because the number of iterations to complete was the same for the sequential and parallel algorithms. Although in Alba (2002), the author does not recommend using that kind of speedup measurement, we have observed in our experiments that the quality of the solutions does not worsen significantly when the stopping criterion is a fixed number of iterations whilst, on the other hand, the speedup achieved is easy to calculate.

4.3 A Generic Parallel Procedure for Dynamic Multi-objective Optimization

The parallel procedure here described is shown in Algorithms 4.1 and 4.2, master and worker, respectively. It is a parallel algorithm for multi-objective optimization that applies an island model along with a master process that divides the population to send sub-populations of the same size to each worker process. For comparison purposes, the parallel algorithm has been generalized in order to be able to run and test different multi-objective evolutionary algorithms.

In this generalized version, every worker searches with the chosen multi-objective evolutionary algorithm (MOEA) the optimal solutions in the search space that has been assigned to it and keeps only those solutions that are not dominated by the others. In this first version, workers share the same search space.

After a fixed number of iterations ($genpar$), the workers send the solutions found to the master, who after gathering all the solutions into a new population, runs an instance of the MOEA (along $genser$ iterations) over the whole population before sending new sub-populations again to the worker processes. The scheme of Figure 4.4 summarizes the way the parallel procedure works.

The use of this generalized parallel dynamic MOEA (pdMOEA) in DMO allows either the execution of more optimization iterations ($genser$ or $genpar$) for a given amount of time (thus exploring more search space and improving the quality of the solutions found), or to speed up the convergence (thus allowing the approach to dynamic problems with higher rates of change).

The EAs are implemented with all the required initialization code outside the main function in order to offer a continuous model of execution, where the population used in the last generation will be intact for the next generation. Furthermore, each MOEA implementation may differ in which sub-population is sent; for example, depending on the implemented algorithm it can be an exact copy of the current population or a copy of the algorithm archive, but for simplicity in Algorithm 4.1 it is represented just as SP_i.

Algorithm 4.1: Master process

```
begin
    Initialize Population of size N
    for $i = 1$ to P workers do
        Send the i-th sub-population, $SP_i$ of size Population/N,
            from Population to the i-th worker
    end
    $t = 1$
    repeat
        for $i = 1$ to P workers do
            Receive the sub-population $SP_i$ from the i-th worker
            Insert $SP_i$ into Population
        end
        Execute the chosen MOEA on Population for
            genser iterations
        if $(t \mod \tau_t) = 0$ then
            Gather statistics of the current time span
        end
        Divide Population among sub-populations $SP_i$
            for each worker $P_i$
        for $i = 1$ to P workers do
            Send the sub-population $SP_i$ to the i-th worker
        end
        $t = t + 1$
    until stop criterion is reached
end
```

4.4 A Fully Distributed Procedure for Dynamic Multi-objective Optimization

It has been seen that the master-worker paradigm provides reasonable levels of speedup, even super-linear ones, but that it suffers from a bottleneck in the communication and processing costs at the master process. Due to this, researchers have tried to develop a fully distributed algorithm for multi-objective optimization.

Algorithm 4.2: Worker process

```
begin
    while true do
        Receive the sub-population from the master process
        Execute the chosen MOEA on sub-population for
          genpar iterations
        Send the sub-population to the master process
    end
end
```

But this task is not as easy to achieve as it could be in other kind of optimization algorithms. The reason is that multi-objective optimization behaves better when the underlying algorithm is searching in a set of tentative solutions at the same time. But due to the fact that in a fully distributed algorithm the processes should work independently, an issue arises on how to redistribute the search space. There are two basic options:

1. Every process uses the whole search space. This option is similar to run the sequential algorithm a number of times equal to the number of workers.

2. Every process explores a given part of the search space.

The second option is the ideal one, because it would allow the best use of the resources avoiding that more than one process is searching on the same area. However, it is also very hard to develop a working procedure that enforces that each process is searching only on a specifically limited and independent area.

There is also a hybrid approach where every process tries to focus on an area while overlapping between processes is allowed but somehow discouraged.

The second approach is fairly possible when a cellular algorithm is employed (Alba et al., 2007), because every process is working on only one solution at any time. But even in this case it is difficult to restrict the search area of every process.

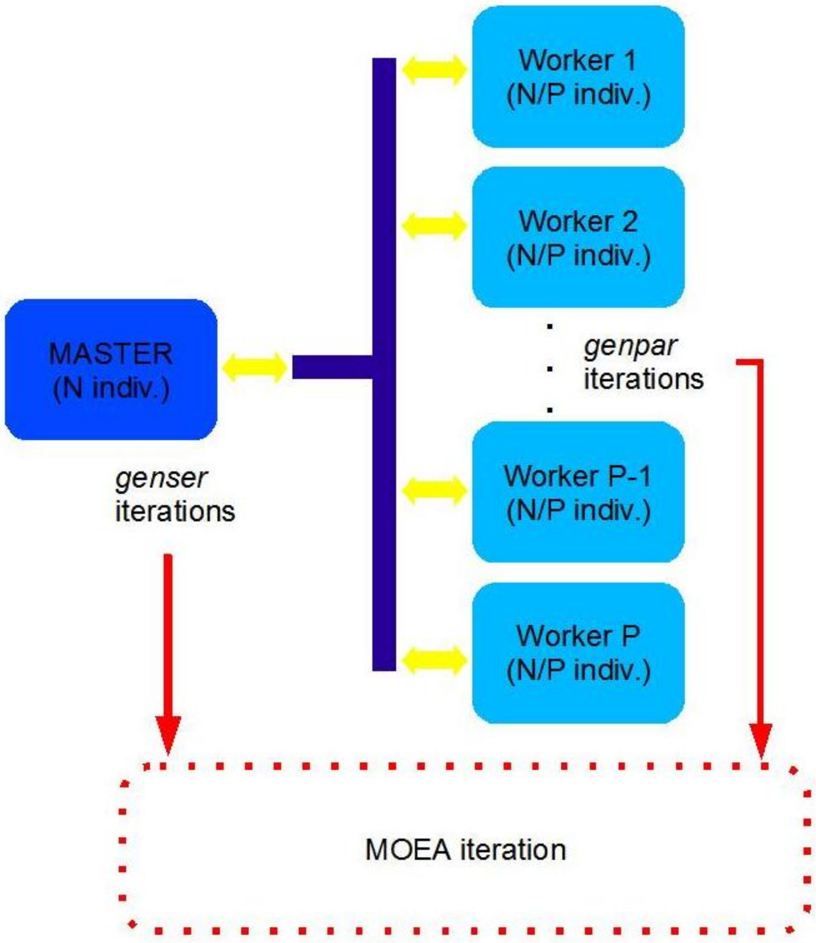

Figure 4.4: Scheme of the pdMOEA operation.

Some researchers have tried to find an elegant and practical way to deal with that mixed approach where processes focus on some part of the search space but some overlapping may occur. Although the overlap should be minimum.

In Deb, Zope, and Jain (2003), the authors propose an island model where while the worker processes search in the decision space, each worker is limited to a different part of the Pareto front. This restriction of the objective space is done implicitly by using a guided domination technique. The main drawback of this approach is that to work properly it requires to know the shape of the multi-objective problem that is to be solved.

Another proposal that provides a MOEA able to search at different space areas is given in Branke, Schmeck, Deb, and Reddy (2004). In it, the authors proposed to divide geometrically the objective search space for every process. The problem behind this solution is that this approach is not good for objective spaces with more than two dimensions.

In Streichert, Ulmer, and Zell (2005), the authors use a similar approach to the one we will show later. They use a clustering algorithm where after a number of generations the MOEA combines the sub-populations and clusters them with the K-means algorithm. Then, the new sub-populations are partitioned and sent to the worker processes. This framework is also similar to the pdMOEA described earlier in Section 4.3 because there is a process where some global computation must be done.

Finally, Lam T. Bui (Bui, 2007; Bui et al., 2009) proposed to distribute the load among the processes by using one adaptative sphere inside every process. The main difference from this framework and the one we will propose later is that in Bui's approach the partition takes place in the decision space whereas in our approach it is done in the objective space. In addition, Bui et al use particle swarm optimization algorithms to guide the spheres in the search space while we use K-means to cluster the solutions.

As it has been said earlier, we tried to find a framework that allowed the processes to work wholly independent from a central master process. This approach can be considered a hybrid of Streichert's and Bui's proposals.

The reason to use a cluster algorithm is that we feel that in this way we could respond to any shape that a problem could have, including discontinuous fronts. This is so important that some researchers even relied on the knowledge of the Pareto front (Deb et al., 2003). Moreover, K-means is a very common clustering algorithm, and its usefulness has been proved in many applications (Navarro and Allen, 1997; Chen, Luo, and Parker, 1998; Kövesi, Boucher, and Saoudi, 2001).

Our proposed distributed algorithm uses a centroid in every process. The centroids are intended to keep distant enough among themselves. Firstly, these centroids must be calculated. In order not to depend on the shape of the Pareto front, a simple calculation is made to place the initial centroid along the axis of the objective space. Examples of these initial centroid locations can be found in Figures 4.5 and 4.6. The calculation of the centroids takes into account objective spaces where the dimensions can differ in range, for example, $f_1(x) \in [0,1]$ and $f_2(x) \in [0,2]$.

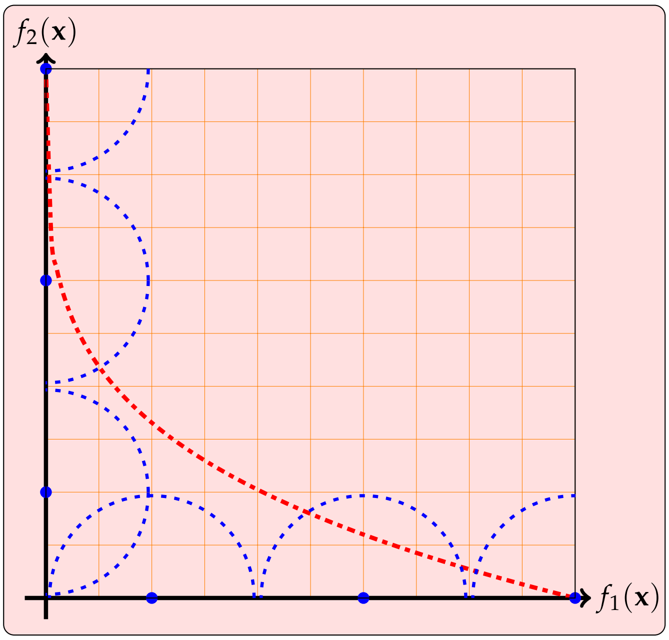

Figure 4.5: Initial location of the centroids for a 2D problem with 6 processes.

Once the initial centroids are calculated, every process allocates a sphere of the same radius. The goal is to evolve these spheres from those shown in Figure 4.5 to those shown in Figure 4.7. Firstly, the radius of the spheres must be adjusted so that every sphere contains some solutions within it. In order to do this, every process runs the selected MOEA for

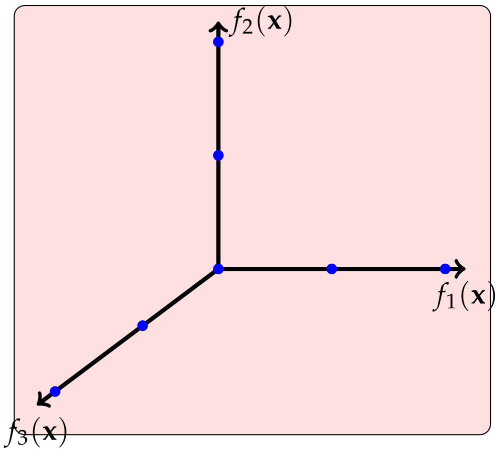

Figure 4.6: Initial location of the centroids for a 3D problem with 7 processes.

a small number of iterations, for example 50, and with the solutions obtained after these 50 iterations have been run, the process is able to adapt the centroid to the nearest solutions with an appropriate radius.

After that, the algorithm is run and the centroid is updated subsequently. This is best illustrated in Algorithm 4.3.

In Line 7, the Output **ND** sentence is meant so that every process provides the current solutions they have found at that time instant. This could be done by sending the solutions found to one process that would gather the solutions from all the running processes and later would print them into a file. However, with the advanced capabilities found in MPI 2, such as distributed I/O it is possible that every process outputs directly their solutions into a file. Certainly, this could determine that some solutions could be dominated by others, but on the other hand, the processes would run at their fastest pace.

4.5 Results

In this section, both proposed procedures are analysed with regard to their results in different test cases.

Algorithm 4.3: Distributed MOEA with K-means

```
Input:
    Iter_initial - The number of initial iterations
Output:
    ND_out - A set of non-dominated solutions
1 Initialize the centroid location
2 Run chosen MOEA for Iter_initial iterations and obtain ND
3 Calculate the centroid radius according to ND
4 repeat
5     Run the chosen MOEA with the current centroid
6     Update the centroid with the last ND
7     Output ND
8 until stopping criterion is reached
```

4.5.1 Results with pdMOEA.

For the sake of the evaluation of DMO algorithms, the FDA set of functions (Farina et al., 2004) has gained the highest relevance among the researchers in this field. In this section, our results are focused only on FDA1 to FDA5. For full details about the FDA functions and the values used in this thesis, please see Section 2.5 in Chapter 2.

The experiments were carried out on an 8-node cluster with two 2 GHz AMD Athlon processors and 2 Gbytes RAM by node, connected via Gigabit Ethernet. The code is implemented in C++ with MPI. SPEA2 and NSGA-II were added anew from the implementations kept in the authors' sites. Experiments that were run in other platforms showed similar results to the ones that are shown in this section (Cámara and Ortega, 2007).

The data shown in the tables has been gathered after running the parallel procedure in 1, 2, 4 and 8 worker processors for each of the MOEAs: SFGA, SFGA2, SPEA2 and NSGA-II; see Chapter 3 for more details about these algorithms. The MOEA parameters were set to the following values:

- master population = 800 individuals

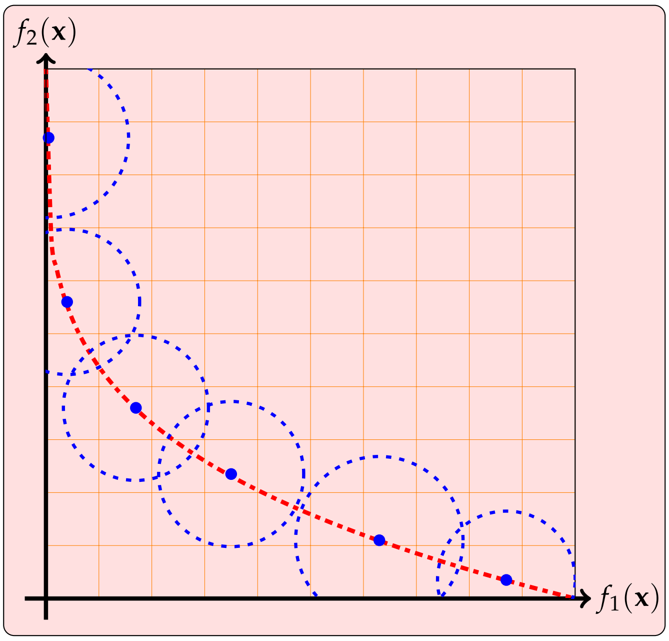

Figure 4.7: Locations of the centroids after some iterations for a 2D problem with 6 processes.

- crowding distance (SFGA and SFGA2), 0.0075
- $\eta_c = 15$
- $\eta_m = 20$
- Mutation probability = 1/(number of decision variables)
- Crossover probability = 0,75
- MOEA iterations: in the workers **genpar** $= 150$ and in the master **genser** $= 50$
- Five runs were made for each algorithm and number of workers

Due to the long running time of some of the MOEAs, only data from $\tau_i = 1$ to $\tau_i = 20$ has been taken into account. τ_i is the parameter for the

FDA functions that says the time instant where the function evaluation takes place. In addition, there is the τ_T parameter which sets the time period in that a function remains stationary. In all our examples, we use $\tau_T = 5$. Thus, every five time instants, there is a change in the current Pareto set (Farina et al., 2004) and the solutions should be recalculated.

Table 4.1: Cumulative time and speedup for FDA1 (pdMOEA)

Algorithm	Workers	Time when τ_i equals			Speedup
		10	15	20	($\tau_i = 20$)
SFGA	1	23,9±0,1	36,6±0,3	49,3 ± 0,3	**1**
	2	22,9±0,6	34,4±0,7	46,3 ± 0,8	**1,1**
	4	14,1±0,2	21,3±0,2	28,5 ± 0,3	**1,7**
	8	8,4±0,2	12,7±0,1	17,0 ± 0,1	**2,9**
SFGA2	1	16,1 ± 0,5	28,8 ± 0,6	44,9 ± 0,8	**1**
	2	11,9 ± 0,4	22,4 ± 0,7	35,2 ± 0,7	**1,3**
	4	11,2 ± 0,2	18,7 ± 0,2	26,3 ± 0,3	**1,7**
	8	7,4 ± 0,1	11,4 ± 0,1	15,5 ± 0,1	**2,9**
SPEA2	1	657,4 ± 4,6	987,6 ± 6,5	1318,3 ± 9,4	**1**
	2	190,2 ± 6,6	285,8 ± 9,9	381,3 ± 13,1	**3,5**
	4	114,8 ± 1,1	172,9 ± 1,7	230,6 ± 2,1	**5,7**
	8	92,7 ± 0,4	139,6 ± 0,6	186,3 ± 0,9	**7,1**
NSGA-II	1	500,9 ± 28,8	742,9 ± 33,1	987,1 ± 52,3	**1**
	2	183,0 ± 12,3	273,1 ± 19,3	364,4 ± 27,6	**2,7**
	4	103,2 ± 3,0	155,1 ± 4,6	208,0 ± 7,6	**4,6**
	8	65,8 ± 1,2	90,9 ± 1,5	121,5 ± 2,2	**8,1**

Table 4.1 shows the cumulative time of the execution of the algorithms for different number of worker processes. The resulting speedups reached by the parallel algorithm is shown in the last column. In order to allow the biggest stability in the algorithms, Table 4.1 reflects the speedup achieved by each algorithm after completing 20 time instants of the function, i.e. $\tau_i = 20$. It can be seen in Table 4.1 that super-linear speedup was achieved for some runs of SPEA2 and NSGA-II.

This behaviour can be explained by the model proposed in Section

4.2. The communication cost modelled in this case is linear with the number of processors, P. As the parallel algorithm allows more diversified populations, it has much to do with the achieved improvement in the performance, particularly with the observed super-linear behaviour.

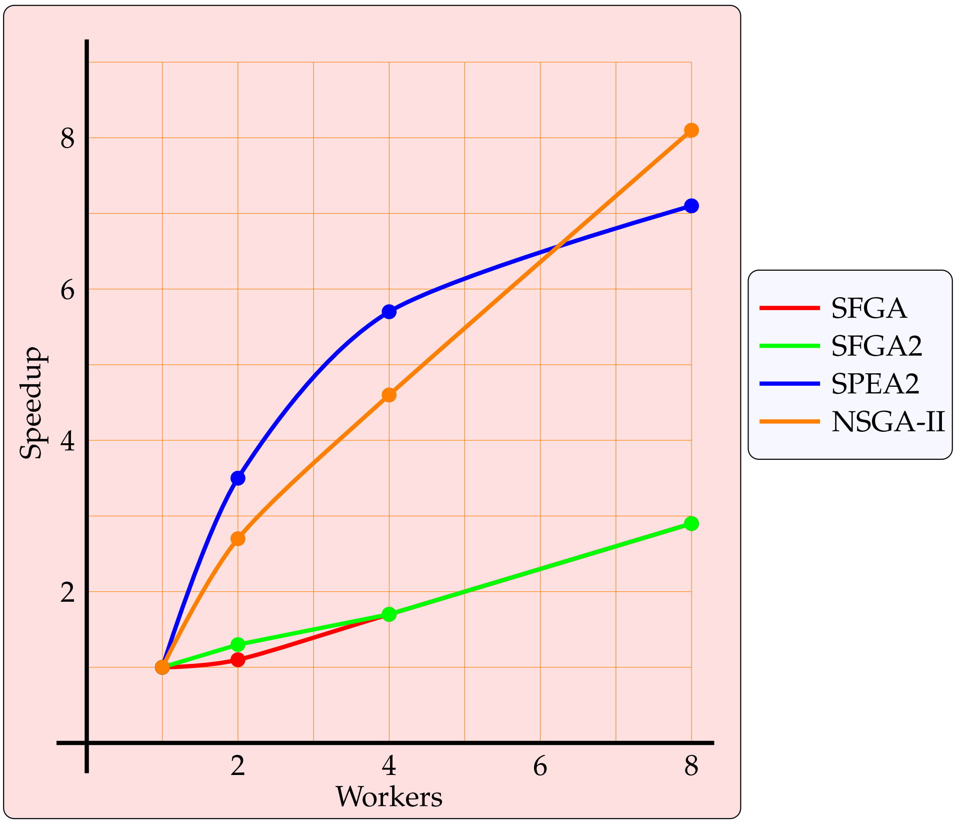

Figure 4.8: Speedup of the different MOEAs.

As it can be seen from Figure 4.8, the speedup curves seem to grow constantly as the number of processors grows. From the speedup model

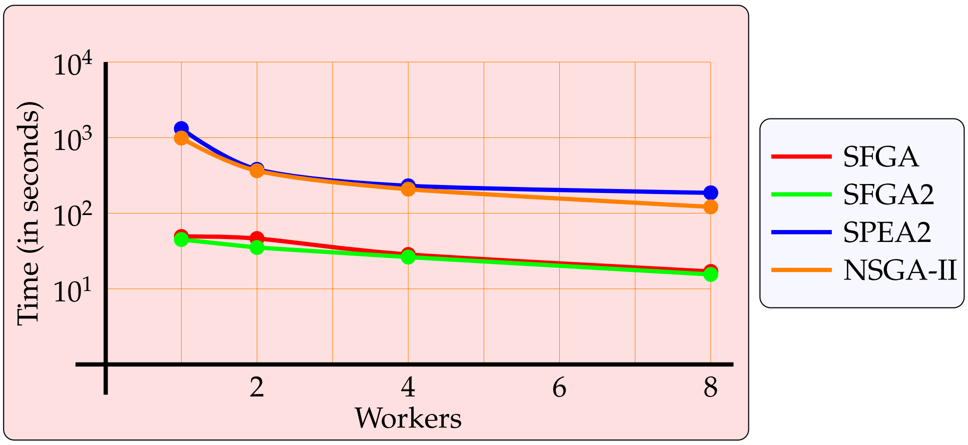

Figure 4.9: Cumulative times of the different MOEAs in semi-log scale.

presented in Section 4.2,

$$
\begin{aligned}
S = & \frac{T_s}{T_p} = \frac{T_s}{T_{p,master} + T_{p,workers} + O(M,P)} \\
= & \frac{gen \times T_{s0}}{(genser \times T_{s0} + genpar \times T_{p0}) + O(M,P)}, \\
& \text{where} \qquad T_{s0} = AMt_0 + BM^r t_1 \\
& \text{and} \qquad T_{p0} = A \times \frac{M}{P} \times t_0 + B \times \left(\frac{M}{P}\right)^r \times t_1.
\end{aligned}
$$

It is clear that, if the speedup tends to a constant as P grows, the function $O(P,M)$ also should tend to a constant. This situation agrees with the communication patterns of our parallel procedure, because although the number of messages between the master and the workers is proportional to the number of processors, P, the size of each message tends to $1/P$ because the master divides the population among the processors.

Figure 4.8 also shows how the speedups for SFGA and SFGA2, which are indeed very close to each other, are not as good as the speedups shown by SPEA2 and NSGA-II. But it should be kept in mind that the cumulative time needed for the execution of NSGA-II and, especially, SPEA2 are, by far, bigger than the time needed by SFGA and SFGA2. This is best seen in Figure 4.9, where a semi-log scale has been used to represent the run times employed by the afore-mentioned algorithms until they fulfilled the execution for $\tau_i = 20$. It shows that the SFGA family algorithms differ in more than one order of magnitude to the times needed by the SPEA2 and NSGA-II algorithms.

Table 4.2 shows the number of non-dominated solutions found by each algorithm. Furthermore, in Table 4.3 there is a compilation of measures consisting of the number of non-dominated solutions found by each algorithm divided by the time needed for that algorithm. In Figure 4.10 there is a plot of these values. This measure cannot be used to indicate whether a certain algorithm is better in terms of quality of the solutions to a multi-objective problem, be it in diversity of the solutions or closeness to the actual Pareto set, but on the other hand, this measure can be useful in DMO. This is because it can indicate a certain advantage of one algorithm over another. The advantage relies on that the superior algorithm could be able to find more solutions per time unit in comparison with the other algorithm. Although this advantage does not imply directly that solutions found by that algorithm had to be better than those found by other algorithms, having more solutions per time unit is a desired feature of any algorithm meant to be used in DMO.

From the Figure 4.10 it is clear that our algorithms SFGA and SFGA2 do not expose super-linear speedups when adding more processors. However, they gave more non-dominated solutions per time unit, which can be seen as an improvement in data throughput instead of time speedup (See Table 2.3). It is worth reminding that in DMO it is common that the algorithm has to meet strict time restrictions, and so, the possibility of having more solutions in less time it is seen as a preferred feature and trade-off over that of having more accurate solutions but at the cost of employing much more time.

Therefore, it is expected that SFGA and SFGA2 can cope with more restrictive time limits without having to reduce the population because they have a smaller runtime and produce more solutions per time unit in comparison to NSGA-II and SPEA2,

In Table 4.4 the quality in terms of the hypervolume (Deb, 2001) of the solutions found by the different algorithms is shown. Hypervolume measures the covered area from a given reference point by the solutions

Table 4.2: Non-dominated solutions for FDA1 (pdMOEA)

Algorithm	Workers	Non-dominated solutions when τ_i equals 10	15	20
SFGA	1	118,4 $\pm$ 3,9	118,0 $\pm$ 4,4	114,8 $\pm$ 4,3
	2	144,8 $\pm$ 5,5	144,4 $\pm$ 6,8	142,2 $\pm$ 4,3
	4	142,0 $\pm$ 5,2	144,6 $\pm$ 2,8	143,4 $\pm$ 1,8
	8	138,4 $\pm$ 2,5	137,0 $\pm$ 2,9	135,6 $\pm$ 2,3
SFGA2	1	147,6 $\pm$ 4,6	235,4 $\pm$ 11,4	310,0 $\pm$ 12,1
	2	280,6 $\pm$ 13,8	385,6 $\pm$ 19,1	436,4 $\pm$ 3,4
	4	290,2 $\pm$ 9,2	322,8 $\pm$ 7,5	320,8 $\pm$ 12,7
	8	225,0 $\pm$ 15.0	231,2 $\pm$ 21.0	230,4 $\pm$ 12,4
SPEA2	1	800,0 $\pm$ 0,0	800,0 $\pm$ 0,0	800,0 $\pm$ 0,0
	2	800,0 $\pm$ 0,0	800,0 $\pm$ 0,0	800,0 $\pm$ 0,0
	4	800,0 $\pm$ 0,0	800,0 $\pm$ 0,0	800,0 $\pm$ 0,0
	8	800,0 $\pm$ 0,0	800,0 $\pm$ 0,0	800,0 $\pm$ 0,0
NSGA-II	1	384,8 $\pm$ 15,8	374,8 $\pm$ 12,6	369,8 $\pm$ 15,1
	2	366,8 $\pm$ 7,5	376,2 $\pm$ 11,2	370,0 $\pm$ 16,7
	4	297,2 $\pm$ 11,1	298,8 $\pm$ 10,4	300,4 $\pm$ 7,2
	8	234,2 $\pm$ 8,6	234,2 $\pm$ 8,1	240,6 $\pm$ 5,4

Table 4.3: Number of non-dominated solutions for each unit of time for FDA1 (pdMOEA)

Workers	SFGA	SFGA2	SPEA2	NSGA-II
1	2,33	6,91	0,61	0,37
2	3,07	12,41	2,10	1,02
4	5,03	12,20	3,47	1,44
8	8,00	14,91	4,30	1,98

found, so in minimizations problems, like the FDA family are, the hypervolume is to be maximized. It can be seen that the best quality was

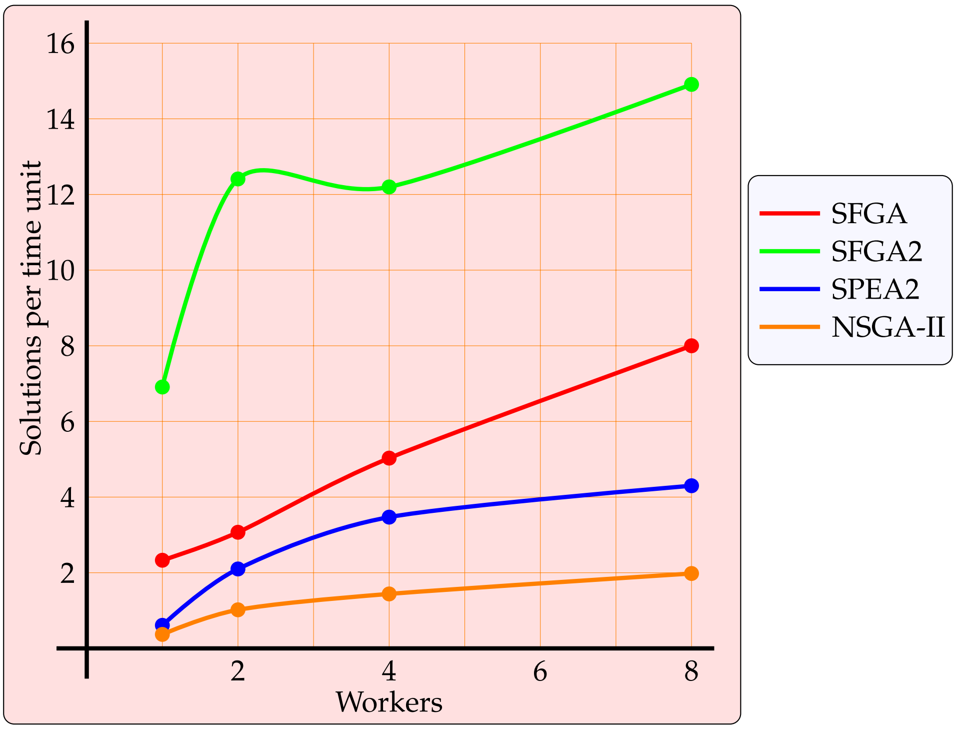

Figure 4.10: Non-dominated solutions per unit of computation time.

attained by the SPEA2 algorithm. This is explained by the fact that this algorithm kept the biggest number of non-dominated solutions, doubling the number of solutions kept by any other algorithm of this study at any moment. Anyway, the four algorithms show very good results in terms of quality, according to the hypervolume indicator. It is important to note that when more worker processes were added the quality did not worse and it even improved for the SFGA and SFGA2 algorithms. However, it cannot be stated that the more workers used, the better the hypervolume.

In what follows Tables 4.5, 4.6, 4.7 and 4.8 are given. Each Table collects, respectively, the Hypervolume, Time, Speedup and number of solutions for each problem, FDA1, FDA2-mod, FDA3-mod, FDA4 and FDA5, and algorithm, SFGA, SFGA2, SPEA2 and NSGA-II.

As it can be seen from Table 4.5 the hypervolume values obtained by the four algorithms for FDA2-mod and FDA3-mod are quite similar with differences smaller than a 3%. However the best values are attained again by SPEA2.

It is more important for our research to have a close look to Table 4.6 where it is shown the time until τ_i for FDA2-mod, FDA3-mod, FDA4 and FDA5 using the four different MOEAs. As it happened earlier for the FDA1 function, the running times of the SFGA family of algorithms is always much smaller than the running times of SPEA2 and NSGA-II algorithms. Two different behaviours can be outlined:

- First of all, for the FDA2-mod and FDA3-mod functions, the SFGA algorithms are at least one order of magnitude faster than SPEA2 and NSGA-II. This is a very important advantage for the SFGA algorithms in comparison to those state-of-the-art MOEAs. In addi-

Table 4.4: Hypervolume for FDA1 (pdMOEA)

Algorithm	Workers	Minimum	Maximum	Average
SFGA	1	0,64	0,64	0,64 ± 0,00
	2	0,65	0,65	0,65 ± 0,00
	4	0,65	0,65	0,65 ± 0,00
	8	0,65	0,65	**0,65** ± **0,00**
SFGA2	1	0,60	0,65	0,63 ± 0,02
	2	0,63	0,66	0,65 ± 0,01
	4	0,64	0,66	0,65 ± 0,01
	8	0,65	0,65	**0,65** ± **0,00**
SPEA2	1	0,66	0,67	0,66 ± 0,01
	2	0,66	0,66	0,66 ± 0,00
	4	0,66	0,66	0,66 ± 0,00
	8	0,66	0,66	**0,66** ± **0,00**
NSGA-II	1	0,65	0,65	0,65 ± 0,00
	2	0,65	0,65	0,65 ± 0,00
	4	0,65	0,65	0,65 ± 0,00
	8	0,65	0,65	**0,65** ± **0,00**

Table 4.5: Hypervolume for FDA2-mod, FDA3-mod, FDA4 and FDA5 (pdMOEA)

Problem	Workers	SFGA	SFGA2	SPEA2	NSGA-II
FDA2-mod	1	0,77 ± 0,04	0,74 ± 0,04	0,83 ± 0,00	0,81 ± 0,01
	2	0,79 ± 0,03	0,77 ± 0,04	0,83 ± 0,00	0,80 ± 0.01
	4	0,80 ± 0,01	0,77 ± 0,05	0,83 ± 0,01	0,80 ± 0,02
	8	0,80 ± 0,02	0,79 ± 0,03	0,83 ± 0,00	0,80 ± 0,02
FDA3-mod	1	3,72 ± 0,02	3,70 ± 0,03	3,75 ± 0,00	3,74 ± 0,01
	2	3,73 ± 0,01	3,70 ± 0,05	3,75 ± 0,00	3,74 ± 0,01
	4	3,73 ± 0,02	3,72 ± 0,02	3,75 ± 0,00	3,74 ± 0,01
	8	3,73 ± 0,01	3,73 ± 0,01	3,75 ± 0,00	3,74 ± 0,01
FDA4	1	1,09 ± 0,08	1,37 ± 0,00	1,37 ± 0,00	1,38 ± 0,00
	2	1,03 ± 0,10	1,37 ± 0,01	1,36 ± 0,01	1,38 ± 0,00
	4	1,08 ± 0,08	1,37 ± 0,00	1,36 ± 0,01	1,38 ± 0,00
	8	1,13 ± 0,03	1,37 ± 0,00	1,36 ± 0,01	1,38 ± 0,00
FDA5	1	5,85 ± 0,09	6,77 ± 0,00	6,03 ± 0,07	6,25 ± 0,00
	2	5,93 ± 0,05	6,77 ± 0,00	5,94 ± 0,13	6,25 ± 0,00
	4	5,85 ± 0,12	6,75 ± 0,01	5,96 ± 0,12	6,25 ± 0,00
	8	5,98 ± 0,03	6,75 ± 0,01	6,07 ± 0,05	6,25 ± 0,00

tion, for these two functions, SPEA2 and NSGA-II are on a par with respect to the running times.

- Moreover, for the three-objective functions FDA4 and FDA5, the SFGA algorithms prove themselves as the fastest algorithms. However, in this case they are only four times faster than SPEA2. It is important to note that NSGA-II behaves very bad in terms of the running time for these functions, and once more it is one order of magnitude slower than the family of SFGA algorithms.

With regard to the speedup, shown in Table 4.7, it can be seen that the results are similar to those that were obtained for the FDA1 function

Table 4.6: Running time in seconds until $\tau_i = 20$ for FDA2-mod, FDA3-mod, FDA4 and FDA5 (pdMOEA)

Problem	Workers	SFGA	SFGA2	SPEA2	NSGA-II
FDA2-mod	1	138,3 ± 2,7	64,67 ± 2,3	3087 ± 36	2351 ± 46
	2	162,0 ± 3,6	46,18 ± 4,1	931,8 ± 31,9	756,1 ± 32,1
	4	79,75 ± 1,2	40,64 ± 2,7	683,7 ± 13,8	422,7 ± 11,3
	8	60,71 ± 1,0	34,25 ± 1,4	629,5± 12,1	455,1 ± 20,1
FDA3-mod	1	125,9 ± 3,6	68,3 ± 2,0	2477 ± 34	2534 ± 50
	2	93,5 ± 1,1	45,9 ± 1,3	888,0 ± 21,0	757,3 ± 31,5
	4	74,9 ± 0,9	38,8 ± 1,6	831,2 ± 15,7	493,7 ± 5,3
	8	58,9 ± 1,0	34,6 ± 0,5	683,9 ± 8,3	631,0 ± 13,5
FDA4	1	641,9 ± 15.1	776,1 ± 7.5	2733 ± 37	23774 ± 89
	2	291,2 ± 8.1	353,1 ± 4.7	1089 ± 13	3793 ± 73
	4	198,7 ± 6.9	238,5 ± 3.1	805,3 ± 9.9	2285 ± 40
	8	169,9 ± 6.2	208,5 ± 3.2	689,6 ± 8.6	2439 ± 53
FDA5	1	746,6 ± 13.8	921,7 ± 8.5	2721 ± 43	22446 ± 71
	2	329,6 ± 9.1	411,0 ± 6.3	1102 ± 17	4001 ± 69
	4	216,7 ± 4.6	285,7 ± 5.2	760,8 ± 8.6	2371 ± 46
	8	189,2 ± 4.3	246,8 ± 4.9	728,0 ± 9.1	2390 ± 44

(Table 4.1). The main differences for these results are:

- The SFGA and SFGA2 algorithms achieve slightly better speedup results than those obtained for FDA1.
- SPEA2 results for FDA4 and FDA5 are almost the same that the ones obtained by SFGA and SFGA2.
- NSGA-II shows super linear speedups for four workers. These are also the maximum peaks of its speedup for all the functions.
- All the algorithms but NSGA-II show increasing linear speedup va-

Table 4.7: Speedup with $\tau_i = 20$ for FDA2-mod, FDA3-mod, FDA4 and FDA5 (pdMOEA)

Problem	Workers	SFGA	SFGA2	SPEA2	NSGA-II
	2	0,85	1,40	3,31	3,11
FDA2-mod	4	1,73	1,59	4,52	5,56
	8	2,28	1,89	4,91	5,17
	2	1,35	1,49	2,79	3,35
FDA3-mod	4	1,68	1,76	2,98	5,13
	8	2,14	1,97	3,62	4,02
	2	2,20	2,20	2,51	6,27
FDA4	4	3,23	3,25	3,39	10,40
	8	3,78	3,72	3,96	9,74
	2	2,26	2,24	2,47	5,61
FDA5	4	3,44	3,23	3,58	9,47
	8	3,95	3,73	3,74	9,39

lues as the number of workers increases.

4.5.2 Analysis of the pdMOEA approach.

After seeing in detail the pdMOEA approach and the results obtained with it, we enumerate the main features that it offers to the researcher with respect to the sequential and other approaches. Among the features of pdMOEA we can see the following:

- It has shown speedup gains, sometimes super linear speedup rates.
- It has obtained better solutions in terms of the hypervolume indicator.
- It offers the possibility of creating hybrid distributed MOEAs where every worker process could run a different MOEA.
- It shows a good level of scalability.

Table 4.8: Number of solutions with $\tau_i = 20$ for FDA2-mod, FDA3-mod, FDA4 and FDA5 (pdMOEA)

Problem	Workers	SFGA	SFGA2	SPEA2	NSGA-II
FDA2-mod	1	105	77	800	268
	2	138	149	800	280
	4	136	197	800	250
	8	125	175	800	185
FDA3-mod	1	77	61	800	252
	2	93	83	799	248
	4	99	144	799	221
	8	106	172	800	212
FDA4	1	800	800	800	981
	2	800	800	800	968
	4	800	800	800	995
	8	800	800	800	993
FDA5	1	800	800	800	1077
	2	800	800	800	1091
	4	800	800	800	1084
	8	800	800	800	1123

- Usually, it provides more solutions per time unit, *throughput*, as more workers are employed.

As it has been said earlier in this thesis, the last of the previously enumerated features is the most important when tackling dynamic problems, because it allows to tackle problems in which the time to solve them is reduced by using more worker processes or choosing a different MOEA to solve them. Of course the last option should be thoroughly studied on a case-by-case way before choosing which algorithm to employ on a given real-world problem.

Unfortunately, this pdMOEA method presents an important disadvantage. This stems from the fact that the workers are using the whole search space at the same time, and a master process is necessary to maintain the global population updated, or put in other way, a master process is required to ensure that all the worker processes have the most up-to-date information about the best global solutions found so far.

Because of this overhead of keeping the constant communication of a master process and its processing times with a global population, a fully distributed procedure has been developed. The next section describes this proposed procedure.

4.5.3 Results with the Fully Distributed Procedure.

The fully distributed procedure described in section 4.4 has been tested with the same MOEAs and test functions as it was tested the pdMOEA approach (Sub-section 4.5.1).

Nevertheless, in the case of a fully distributed procedure, the tests have been carried out only for four and eight workers processes. The reason behind this is that the pdMOEA+ procedure must show an excellent behaviour when many workers are involved and so, it is not relevant the performance of the algorithm with only two workers. Because of that only results for 4 and 8 workers are reproduced in Tables 4.9 and 4.10.

The tests were made with the same values as the the pdMOEA procedure but the new parameter *radius* of the sphere was added. For completeness sake, they are reproduced here again:

- master population = 800 individuals
- crowding distance (SFGA and SFGA2), 0.0075
- $\eta_c = 15$
- $\eta_m = 20$
- Mutation probability = 1/(number of decision variables)
- Crossover probability = 0,75
- MOEA iterations: in the workers **genpar** $= 150$ and in the master **genser** $= 50$
- Radius = 0,35
- Five runs were made for each algorithm and number of workers

Unfortunately, the obtained results were not as good as expected. In terms of the quality of the obtained solutions, the pdMOEA+ results have been clearly worse than those results given by pdMOEA. pdMOEA+ also produces less number of solutions for some combinations of algorithms and number of workers, while it provides more solutions for other combinations. The numbers of solutions for each problem and MOEA are collected in Table 4.9.

With respect to the speedup (see Table 4.10), the pdMOEA+ approach shows better results than the pdMOEA approach showed (Table 4.6). Nonetheless, the improvement shown by these results should be taken into account with care. The reason is that, as it has been said before, for some of the tests, the number of solutions has decreased with respect to the pdMOEA and to the sequential run of the algorithms. Thus, if the MOEAs are producing less solutions, they need less time to compute them.

Therefore, it can be said that these results do not show enough improvement in the quality of the solutions nor in speedup. The reason behind these awkward results is that the processes tend towards very

Table 4.9: Number of non-dominated solutions when $\tau_i = 20$ for FDA1, FDA2-mod, FDA3-mod, FDA4 and FDA5 (pdMOEA+)

Problem	Workers	SFGA	SFGA2	SPEA2	NSGA-II
FDA1	4	138 ± 5	226 ± 7	30 ± 1	251 ± 4
	8	313 ± 6	289 ± 6	27 ± 1	308 ± 4
FDA2-mod	4	111 ± 6	122 ± 3	21 ± 1	203 ± 3
	8	120 ± 3	109 ± 3	42 ± 1	270 ± 4
FDA3-mod	4	96 ± 3	105 ± 5	36 ± 2	192 ± 3
	8	257 ± 5	292 ± 6	41 ± 2	311 ± 4
FDA4	4	540 ± 8	67 ± 3	64 ± 2	73 ± 3
	8	288 ± 4	107 ± 3	91 ± 2	68 ± 3
FDA5	4	153 ± 4	126 ± 3	55 ± 2	166 ± 4
	8	216 ± 5	222 ± 4	73 ± 2	164 ± 4

narrow areas. This suggests that some more knowledge should be added to the algorithm in order to acquire a more generic behaviour, as that shown in Bui's results (Bui et al., 2009).

4.6 Summary

The main conclusions extracted from this chapter are the following ones:

1. A generic parallel procedure, pdMOEA, has been proposed and tested. Results with pdMOEA have shown that some MOEAs obtain almost super-linear speedups, while different MOEAs provide different speedup figures and solutions per time unit, throughput. As it has been said multiple times along this thesis and this chapter, these two features are desired when dealing with real-world dynamic optimization problems because the practitioner and the researcher are both interested in getting the best possible solutions

Table 4.10: Time to reach $\tau_i = 20$ for FDA1, FDA2-mod, FDA3-mod, FDA4 and FDA5 (pdMOEA+)

Problem	Workers	SFGA	SFGA2	SPEA2	NSGA-II
FDA1	4	14,1 ± 0,7	8,7 ± 0,5	39,2 ± 1,3	67,6 ± 1,7
	8	4,9 ± 0,3	4,8 ± 0,3	9,4 ± 0.4	24,4 ± 0.9
FDA2-mod	4	18,3 ± 0,6	9,0 ± 0,5	38,9 ± 1,1	53,8 ± 1,2
	8	12,8 ± 0,5	6,9 ± 0,4	27,5 ± 0,8	42,0 ± 0.9
FDA3-mod	4	15,7 ± 0,4	9,0 ± 0,4	91,7 ± 1,3	59,1 ± 1,0
	8	16,1 ± 0,5	12,8 ± 0,4	27,0 ± 0,7	45,9 ± 0,6
FDA4	4	18,6 ± 0,5	27,8 ± 0,5	42,6 ± 0,8	195,2 ± 2,1
	8	16,3 ± 0,5	21,6 ± 0,8	29,8 ± 0,9	120,9 ± 1,4
FDA5	4	20,5 ± 0,9	31,3 ± 1,0	36,9 ± 0,7	177,6 ± 1,8
	8	17,2 ± 0,6	23,3 ± 0,8	30,8 ± 0,5	45,7 ± 0,9

within restrictive time limits. SFGA and SFGA2 have proven to be able to accomplish these objectives, whilst other MOEAs such as SPEA2 and NSGA-II require considerable more time to improve slightly the quality of the obtained solutions. The most salient feature of pdMOEA when used with SFGA2 is that gives very fast results with a quality not much worse than the obtained quality with SPEA2 or NSGA-II, and in addition, it produces many solutions per time unit, allowing to solve a DMO problem with a very good representation of the Pareto front. Specifically, pdMOEA has shown super-linear speedup when used with SPEA2 and NSGA-II, and a smaller speedup for SFGA and SFGA2. In addition, the quality of the solutions has improved when more than one process were employed to solve the problems. Finally, all the algorithms have shown increases in the number of solutions per time unit or throughput as the number of processes grew.

2. In addition, pdMOEA+, a procedure aimed at fully distributed computation, has been proposed and tested. The first results with this new approach have shown a promising future while it has also shown that more development should be done in order to achieve better results that can rival with those provided by the pdMOEA approach or sequential MOEAs. Concretely, pdMOEA+ has shown faster execution than pdMOEA. Moreover, it offers linear speedup and produces a good number of solutions per unit time. On the other hand, the quality of the solutions given has been clearly worse than the quality obtained by pdMOEA. Future research for pdMOEA+ should look for a way to keep, and even to improve, the time responses that it has shown while not sacrificing the quality nor the number of the solutions.

The research shown in this Chapter has been published, partly or fully, in Cámara et al. (2007a,b, 2009b, 2008c,b, 2010).

The next, and last chapter, summarizes the whole thesis and its main conclusions. In addition, it lists the contributions that have arisen from this thesis.

"Science never solves a problem without creating ten more."

George Bernard Shaw

5 Summary Conclusions and Contributions

THE conclusions from this PhD thesis are collected in this chapter, along with the main contributions that were produced while researching on this thesis. Moreover, different lines of future work that could extend on what this thesis presents are summarized in the last part of the chapter.

5.1 Conclusions and Contributions

The main aim of this PhD thesis has been to study the usefulness of parallel processing methods to solve dynamic multi-objective optimization problems by using multi-objective evolutionary algorithms. From the research work done on such parallel processing topics some contributions have been made to different research areas, that although related are independent among themselves. These research areas are:

- Multi-objective evolutionary algorithms,

- Dynamic multi-objective optimization, and
- The use of parallel processing of MOEAs to solve DMO problems.

The contributions that stem from the work presented in this PhD thesis are enumerated and commented in what follows:

- An **analysis** and a detailed **description of the performance measures** that should be used with DMO problems have been given.
- **Performance measures** for dynamic and multi-objective optimization problems when the **Pareto fronts are known** have been adapted and proposed from performance measures for dynamic single-objective optimization problems. These measures are intended to assess the accuracy, stability and reaction time of a given algorithm when solving a DMO problem. But these measures present a serious practical shortcoming, because they can only be used with those problems for which we know the Pareto front beforehand.
- **Performance measures** for dynamic and multi-objective optimization problems when the **Pareto fronts are unknown** have been also introduced. These measures are also aimed at the accuracy, stability and reaction time of an algorithm, but contrary to the earlier ones, these new measures work with any algorithm that is used for solving any problem for which we do not know its Pareto fronts. Thus, it overcomes the drawback of the first measures, enabling researchers to use these performance measures with complex test cases and real world problems.
- **Improvements for two of the five standard test cases** for DMO problems (Farina et al., 2004) have been given. FDA2-mod and FDA3-mod functions are modified versions of the original FDA2 and FDA3 ones. The modifications solve some underlying problems in the original functions and allow the use of these two new test cases in the development of algorithms for DMO problems.
- **SFGA2, a *low-cost* MOEA** has been introduced. It is a very fast MOEA which works only with a single front of non-dominated

solutions. It shows a very fast running time while providing sets of solutions whose average quality is comparable to the quality of those solutions obtained by state-of-the-art MOEAs such as SPEA2 and NSGA-II.

- Due to the SFGA2 very low running time in comparison with other MOEAs, **SFGA2 is an ideal option to cope with dynamic problems**, while they also have to meet demanding time restrictions.
- **SFGA2 improves the performance** of SFGA in problems with more than two objectives.
- **SFGA2 produces a big number of solutions per time unit or throughput**, which is a must-have feature for MOEAs that are to be used to solve DMO problems.
- An **analysis theory to study parallel procedures** for MOEAs has also been described.
- pdMOEA, a **parallel procedure** has been introduced in order to solve DMO problems by using MOEAs. It is a **hybrid between master-worker and island approaches**.
- pdMOEA shows very good results in comparison with the sequential approach, specifically, it provides **super-linear speedup** with those MOEAs that required a longer execution time such as SPEA2 and NSGA-II.
- **pdMOEA improves the quality of the solutions** obtained by the different MOEAs when more than one process is employed.
- In addition, for the algorithms SFGA and SFGA2 where the obtained speedup is below super-linearity, **pdMOEA increases the number of solutions per time unit, throughput**, which is a desired feature for algorithms solving DMO problems.
- The combination of **pdMOEA and SFGA2 rival with the results given by NSGA-II and SPEA2** while requiring only a fraction of their execution time. Nevertheless, it presents some shortcomings that arise from the inner working of multi-objective optimization

that requires a global knowledge of the population in order to avoid the repetition of the same calculations in different workers.

- A **island pure model** has been proposed. This model limits the search area of each worker process by using a K-means clustering algorithm. Although the experimental results obtained so far have not shown its superiority to pdMOEA, there are signs that this island K-means based procedure could improve further the good results obtained so far with pdMOEA.

5.2 Publications

The results of the research work presented in this PhD thesis have been published. In what follows these publications are listed, grouped by type of publication and sorted chronologically within each group:

1. **Journals**

    - *High performance computing for dynamic multi-objective optimisation*, High Performance Systems Architecture 1 (2008), no. 4, pp 241–250, Mario Cámara, Julio Ortega and Francisco de Toro.
    - *A single front genetic algorithm for parallel multi-objective optimization in dynamic environments*, Neurocomputing 72 (2009), 3570–3579, Mario Cámara, Julio Ortega and Francisco de Toro.

2. **Chapters**

    - *Approaching Dynamic Multi-objective Optimization Problems by Using Parallel Evolutionary Algorithms* in Advances in multi-objective nature inspired computing, Studies in Computational Intelligence, vol. 272, pp. 59–80, Springer, 2010, Mario Cámara, Julio Ortega and Francisco de Toro. This is a chapter by-invitation only.

3. **International Conferences**

- *Parallel processing for multi-objective optimization in dynamic environments*, Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS '07), 2007, pp. 1–8, Mario Cámara, Julio Ortega and Francisco de Toro.
- *The parallel single front genetic algorithm (PSFGA) for dynamic multi-objective optimization*, Proceedings Of The 9th International Work-Conference on Artificial Neural Networks (IWANN '07) (F. Sandoval et al., eds.), LNCS, no. 4507, Springer-Verlag, 2007, pp. 300–307, Mario Cámara, Julio Ortega and Francisco de Toro.
- *A diversity enhanced single front multiobjective algorithm for dynamic optimization problems*, Proceedings of the 1st International Conference on Metaheuristics and Nature Inspired Computing (META'08), 2008, Mario Cámara, Julio Ortega and Francisco de Toro.
- *Parallel multi-objective optimization evolutionary algorithms in dynamic environments*, Proceedings of The First International Workshop on Parallel Architectures and Bioinspired Algorithms (Juan Lanchares, Francisco Fernández, and José L. Risco-Martín eds.), vol. 1, 2008, pp. 13–20, Mario Cámara, Julio Ortega and Francisco de Toro.
- *Performance measures for dynamic multi-objective optimization*, Proceedings of the 10th International Work-Conference on Artificial Neural Networks (IWANN '09) (J. Cabestany, Alberto Prieto and Francisco Sandoval, eds.), Lecture Notes in Computer Science, vol. 5517, Springer-Verlag, June 2009, pp. 760 – 767, Mario Cámara, Julio Ortega and Francisco de Toro.

4. **National Conferences**

- *Procesamiento paralelo para optimización multiobjetivo en entornos dinámicos*, XVII Jornadas De Paralelismo, 2006, Mario Cámara, Julio Ortega and Francisco de Toro.
- *Un algoritmo paralelo de frente único para optimización multiobjetivo dinámica*, I Jornadas de Algoritmos Evolutivos y Meta-

heurísticas, 2007, pp. 113–120, Mario Cámara, Julio Ortega and Francisco de Toro.

- *Medidas de rendimiento para optimización dinámica multiobjetivo*, Actas del VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'09), 2009, pp. 357 – 364, Mario Cámara, Julio Ortega and Francisco de Toro.

5. **Other**

- *Optimización multiobjetivo paralela en entornos dinámicos*, Tech. report, Dept. de Arquitectura y Tecnología de Computadores, Universidad de Granada, June 2007. Mario Cámara.
- *Parallel evolutionary algorithms for dynamic multiobjective optimization*, HPC-Europa report 2007 (Paola Alberigo, Giovanni Erbacci, Francesca Garofalo, and Silvia Monfardini, eds.), vol. 1, CINECA, 2007, pp. 334–337. Mario Cámara and Julio Ortega.

5.3 Future Work

While working on this PhD thesis, some areas to improve further have arisen. They form the basis for future work and are listed in what follows:

- **Improvements to SFGA2**. The algorithm has still room for improvements such as finding even better solutions without compromising its main feature of being a very fast approach. This could be achieved by fine tuning how the algorithm selects the solutions to carry onto the next generation. In addition to improvements in the quality of the solutions, changes could also be introduced to produce an even faster algorithm.

- **Improvements to pdMOEA**. The pdMOEA approach could be used to run parallel MOEA processes, where each process could be able to run a different MOEA. This is a straightforward addition to the pdMOEA procedure but it would require some dynamical adjustments in order to work seamlessly with MOEAs that have different run times. For example, if we want to run two processes with SFGA2 and other two with NSGA-II, the number of iterations for each one should be different. The correct number for each process should be calculated dynamically every time the algorithm is run.

- **Improvements to pdMOEA+**. Obviously, the results shown by pdMOEA+ have not been so positive as expected. As it has been noted at the end of Chapter 4, changes should be done to pdMOEA+ so that it can produce solutions as good as the ones provided by pdMOEA. After achieving this improvement on the quality of the solutions, pdMOEA+ could be easily transformed into a procedure that would allow to run different MOEAs in every process. In contrast to pdMOEA, this is easier to accomplish by pdMOEA+ because it does not require any communication across a central master process. Moreover, some considerations, from those that appear in Bui, Essam, and Hussein A. Abbass (2010), should also be taken into account when improving the pdMOEA+ approach.

- **Comparison with the cellular approach**. It would be interesting to implement a cellular approach to solve DMO problems. Once it is implemented it could be used to compare this new approach with both pdMOEA and pdMOEA+. From the results obtained in this comparison, it would be easy to infer which of the three approaches is the most suitable one to use with DMO problems.

- **Comparisons with parallel approaches based on other metaheuristics**. Another useful task to accomplish in the future it is to carry a series of side-by-side comparisons of pdMOEA and pdMOEA+ using SFGA2, SPEA2 and NSGA-II with other parallel approaches that use other kind of metaheuristics such as Particle Swarm Optimization, Ant Colony Optimization, Artificial Immune Systems, etc.

This chapter has been dedicated to review the main contributions that have been produced as a result of the work done for this thesis about providing an effective parallel procedure to solve DMO problems by using multi-objective evolutionary algorithms. In addition, it has been shown the different lines of future work that can extend further the work here presented. In this way, this chapter brings this thesis to an end.

"La ciencia puede imponer límites al conocimiento, pero no debería imponerlos a la imaginación"
Bertrand Russell

A
Conclusiones Finales y Aportaciones

LAS conclusiones de esta memoria se resumen en este apéndice, conjuntamente con las principales contribuciones que se han producido como fruto de la investigación objeto de esta tesis. Además, en la última parte del apéndice, se ofrecen diferentes líneas de investigación de trabajo futuro que podrían extender lo que se ha presentado en los capítulos anteriores.

A.1 Conclusiones y Aportaciones

El objetivo principal de esta tesis doctoral ha sido el desarrollo de un método de procesamiento paralelo para la resolución de problemas de optimización dinámica multiobjetivo mediante el uso de algoritmos evolutivos multiobjetivo. Durante la investigación realizada con el fin de conseguir tal procedimiento paralelo se han realizado algunas contribuciones a diferentes áreas de investigación que, aunque están relacionadas, son independientes entre sí. Estas áreas de investigación son:

- Los algoritmos evolutivos multiobjetivo (MOEA),

- La optimización dinámica multiobjetivo (DMO), y
- El uso del paralelismo junto a algoritmos evolutivos multiobjetivo para resolver problemas de optimización dinámica multiobjetivo.

Las contribuciones más relevantes que han surgido del trabajo presentado en esta tesis doctoral son:

- Se han propuesto un **análisis** y **una descripción detallada** de las medidas de rendimiento que se deberían usar con problemas DMO.
- Se han adaptado y propuesto **medidas de rendimiento** para problemas de optimización dinámica multiobjetivo **cuando los frentes de Pareto son conocidos**. El propósito de estas medidas es conocer la precisión, estabilidad y el tiempo de reacción de un algoritmo dado al resolver un problema dinámico multiobjetivo. Estas medidas presentan un grave problema práctico, ya que solo se pueden utilizar con aquellos problemas en los que se conocen los frentes de Pareto por adelantado.
- Se han introducido también **medidas de rendimiento** para problemas de optimización dinámica multiobjetivo **cuando los frentes de Pareto son desconocidos**. Estas medidas están orientadas también a la precisión, estabilidad y el tiempo de reacción de un algoritmo, pero al contrario de las medidas anteriores, estas nuevas medidas funcionan con cualquier algoritmo que resuelva un problema para el que no se conozcan sus frentes de Pareto. De este modo, soluciona el inconveniente de las primeras medidas, permitiendo a las investigadores usar estas nuevas medidas con problemas de prueba complejos y otros tomados del mundo real.
- Se han realizado **mejoras a dos de los cinco casos de prueba** estándar para problemas DMO (Farina et~al., 2004). Las funciones FDA2-mod y FDA3-mod son versiones modificadas de las originales FDA2 y FDA3. Estas modificaciones resuelven algunos problemas subyacentes en las funciones originales. Esto permite usar estos dos casos de prueba en el desarrollo de algoritmos para problemas DMO.

- Se ha introducido **SFGA2, un algoritmo *de bajo coste***. SFGA2 mejora el rendimiento de SFGA en problemas con más de dos objetivos. Además, mantiene un tiempo de ejecución muy rápido a la par que produce conjuntos de soluciones cuya calidad media es comparable a la de las soluciones obtenidas con algoritmos punteros como SPEA2 y NSGA-II.

- **SFGA2 mejora el rendimiento** obtenido con SFGA **para problemas con más de dos objetivos**.

- A causa de su reducido tiempo de ejecución en comparación con otros algoritmos evolutivos multiobjetivo, **SFGA2 es una opción ideal para afrontar problemas dinámicos**, a la vez que se cumplen exigentes restricciones de tiempo.

- Además, **SFGA2 produce un gran número de soluciones por unidad de tiempo**, lo que es una cualidad necesaria para los algoritmos evolutivos multiobjetivo que se vayan a usar en la resolución de problemas DMO.

- Se ha descrito una **herramienta teórica para estudiar procedimientos paralelos** para algoritmos evolutivos multiobjetivo.

- Se ha introducido pdMOEA, un **procedimiento paralelo que es un híbrido entre los enfoques maestro-trabajador y el modelo de islas**, para resolver problemas por medio de algoritmos evolutivos multiobjetivo.

- pdMOEA ha mostrado muy buenos resultados en comparación con el enfoque secuencial. Concretamente, ha dado ganancias supralineales con aquellos MOEA que requirieron mayor tiempo de ejecución como fueron SPEA2 y NSGA-II.

- **pdMOEA mejora la calidad de las soluciones** encontradas cuando se utiliza para resolver el problema más de un proceso.

- Además, para los algoritmos SFGA y SFGA2 donde la ganancia obtenida está por debajo de la supralinealidad, **pdMOEA aumenta el número de soluciones por unidad de tiempo o rendimiento**, throughput en inglés, lo que es una cualidad deseada para los algoritmos que resuelven problemas dinámicos.

- La combinación de **pdMOEA con SFGA2 compite con los resultados obtenidos por NSGA-II y SPEA2** a la vez que requiere solo una fracción del tiempo de ejecución de los otros. No obstante, pdMOEA presenta algunas deficiencias que provienen del funcionamiento interno de los algoritmos evolutivos multiobjetivo que requieren un conocimiento global de la población para evitar la repetición del mismo trabajo en diferentes trabajadores.
- Se ha propuesto un **modelo basado en islas** llamado pdMOEA+. Este modelo limita el área de búsqueda de cada proceso trabajador usando un algoritmo de K-medias. Aunque los resultados experimentales obtenidos de momento no han mostrado su superioridad a pdMOEA, existen signos de que el procedimiento basado en islas y K-medias podrá mejorar más aún los buenos resultados obtenidos hasta ahora por pdMOEA.

A.2 Principales Aportaciones

Los resultados de la investigación presentada en esta memoria han dado lugar a diferentes publicaciones. Dichas publicaciones se muestran a continuación, agrupadas por tipo y ordenadas cronológicamente dentro de cada grupo:

1. **Revistas internacionales**
    - *High performance computing for dynamic multi-objective optimisation*, High Performance Systems Architecture 1 (2008), no. 4, pp 241–250, Mario Cámara, Julio Ortega y Francisco de Toro.
    - *A single front genetic algorithm for parallel multi-objective optimization in dynamic environments*, Neurocomputing 72 (2009), 3570–3579, Mario Cámara, Julio Ortega y Francisco de Toro.
2. **Capítulos de libros**

- *Approaching Dynamic Multi-objective Optimization Problems by Using Parallel Evolutionary Algorithms* in Advances in multi-objective nature inspired computing, Studies in Computational Intelligence, vol. 272, pp. 59–80, Springer, 2010, Mario Cámara, Julio Ortega y Francisco de Toro. Capítulo publicado solo por invitación para participar en el libro.

3. **Congresos internacionales**

- *Parallel processing for multi-objective optimization in dynamic environments*, Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS '07), 2007, pp. 1–8, Mario Cámara, Julio Ortega y Francisco de Toro.
- *The parallel single front genetic algorithm (PSFGA) for dynamic multi-objective optimization*, Proceedings Of The 9th International Work-Conference on Artificial Neural Networks (IWANN '07) (F. Sandoval et al., eds.), LNCS, no. 4507, Springer-Verlag, 2007, pp. 300–307, Mario Cámara, Julio Ortega y Francisco de Toro.
- *A diversity enhanced single front multiobjective algorithm for dynamic optimization problems*, Proceedings of the 1st International Conference on Metaheuristics and Nature Inspired Computing (META'08), 2008, Mario Cámara, Julio Ortega y Francisco de Toro.
- *Parallel multi-objective optimization evolutionary algorithms in dynamic environments*, Proceedings of The First International Workshop on Parallel Architectures and Bioinspired Algorithms (Juan Lanchares, Francisco Fernández y José L. Risco-Martín eds.), vol. 1, 2008, pp. 13–20, Mario Cámara, Julio Ortega y Francisco de Toro.
- *Performance measures for dynamic multi-objective optimization*, Proceedings of the 10th International Work-Conference on Artificial Neural Networks (IWANN '09) (J. Cabestany, Alberto Prieto y Francisco Sandoval, eds.), Lecture Notes in Computer Science, vol. 5517, Springer-Verlag, June 2009, pp. 760 – 767, Mario Cámara, Julio Ortega y Francisco de Toro.

4. **Congresos nacionales**

   - *Procesamiento paralelo para optimización multiobjetivo en entornos dinámicos*, XVII Jornadas De Paralelismo, 2006, Mario Cámara, Julio Ortega y Francisco de Toro.
   - *Un algoritmo paralelo de frente único para optimización multiobjetivo dinámica*, I Jornadas de Algoritmos Evolutivos y Metaheurísticas, 2007, pp. 113–120, Mario Cámara, Julio Ortega y Francisco de Toro.
   - *Medidas de rendimiento para optimización dinámica multiobjetivo*, Actas del VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'09), 2009, pp. 357 – 364, Mario Cámara, Julio Ortega y Francisco de Toro.

5. **Otros**

   - *Optimización multiobjetivo paralela en entornos dinámicos*, Memoria del Diploma de Estudios Avanzados, Dept. de Arquitectura y Tecnología de Computadores, Universidad de Granada, June 2007. Mario Cámara.
   - *Parallel evolutionary algorithms for dynamic multiobjective optimization*, HPC-Europa report 2007 (Paola Alberigo, Giovanni Erbacci, Francesca Garofalo y Silvia Monfardini, eds.), vol. 1, CINECA, 2007, pp. 334–337. Mario Cámara y Julio Ortega.

A.3 Trabajo Futuro

Durante la realización de esta tesis doctoral, han surgido algunas áreas de mejora. En ellas se basan las líneas de trabajo futuro, que se listan a continuación:

- **Mejoras a SFGA2**. El algoritmo aún tiene lugar para mejoras tales como encontrar aún mejores soluciones sin comprometer su principal cualidad de ser un enfoque muy rápido. Esto se podría conseguir ajustando la manera en que el algoritmo selecciona las soluciones que llevará a la siguiente generación. Además, a las mejoras en la calidad de las soluciones, se podrían introducir cambios para producir un algoritmo todavía más rápido.

- **Mejoras a pdMOEA**. El enfoque de pdMOEA podría usado para procesos MOEA paralelos, donde cada proceso podría ejecutar un MOEA diferente. Esta es una adición sencilla al procedimiento pdMOEA que sin embargo requeriría algunos ajustes dinámicos para trabajar sin variaciones apreciables en los MOEA que poseen tiempos de ejecución diferentes. Por ejemplo, si queremos ejecutar dos procesos con SFGA2 y otros dos con NSGA-II, el número de iteraciones para cada uno debería ser diferente. El número correcto para cada proceso debería calcularse dinámicamente cada vez que el algoritmo se ejecuta.

- **Mejoras a pdMOEA+**. Obviamente, los resultados ofrecidos por pdMOEA+ no han sido tan buenos como se esperaban. Como se ha dicho al final del Capítulo 4, habría que realizar cambios a pdMOEA+ para que pueda producir soluciones tan buenas como las producidas por pdMOEA. Después de conseguir esta mejora en la calidad de las soluciones, se podría transformar fácilmente pdMOEA+ en un procedimiento que permitiera ejecutar diferentes MOEA en cada proceso. Al contrario que pdMOEA, es más fácil para pdMOEA+ porque no requiere comunicación a través del proceso maestro. Además, deberían tenerse en cuenta algunas consideraciones de las que aparecen en Bui et~al. (2010), al realizar las mejoras sobre el enfoque pdMOEA+.

- **Comparación con el enfoque celular**. Sería interesante implementar un enfoque celular para resolver problemas DMO. Una vez sea implementado, podría usarse para comparar este enfoque celular con pdMOEA y pdMOEA+. De los resultados obtenidos en esta comparación, sería fácil inferir cuales de los tres enfoques es el más adecuado para usar con problemas DMO.

- **Comparación con enfoques paralelos basados en otras metaheurísticas**. Otra tarea útil para afrontar en el futuro consiste en realizar una serie de comparaciones de pdMOEA y pdMOEA+ usando SFGA2, SPEA2 y NSGA-II con enfoques paralelos que usen otro tipo de metaheurísticas tales como optimización de enjambres de partículas, optimización de colonias de hormigas, sistemas inmunes artificiales, etc.

List of Figures

List of Tables

List of Algorithms

Bibliography

ALBA, E. 1999. Análisis y diseño de algoritmos genéticos paralelos distribuidos. Ph.D. thesis, University of Málaga.

ALBA, E. 2002. Parallel evolutionary algorithms can achieve super-linear performance. *Inf. Process. Lett. 82,* 1, 7–13.

ALBA, E. 2005. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley.

ALBA, E., DORRONSORO, B., LUNA, F., NEBRO, A. J., BOUVRY, P., AND HOGIE, L. 2007. A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan manets. *Comput. Commun. 30,* 4, 685–697.

AUGER, A., BADER, J., BROCKHOFF, D., AND ZITZLER, E. 2009. Theory of the Hypervolume Indicator: Optimal μ-Distributions and the Choice of the Reference Point. In *Foundations of Genetic Algorithms (FOGA 2009)*. ACM, 87–102.

BARRICELLI, N. A. 1957. Symbiogenetic evolution processes realized by artificial methods. *Methodos 9,* 35–36, 143 – 182.

BARRICELLI, N. A. 1963a. Numerical testing of evolution theories. Part I. Theroetical introduction and basic tests. *Acta Biotheoretica 16,* 1–2, 69–98.

BARRICELLI, N. A. 1963b. Numerical testing of evolution theories. Part II. Preliminary tests of performance. Symbiogenesis and terrestrial life. *Acta Biotheoretica 16,* 3–4, 99–126.

BAÑOS, R., GIL, C., PAECHTER, B., AND ORTEGA, J. 2006. Parallelization of population-based multi-objective metaheuristics: An empirical study. *Applied Mathematical Modelling 30,* 7, 578–592.

BAÑOS, R., GIL, C., PAECHTER, B., AND ORTEGA, J. 2007. A hybrid meta-heuristic for multi-objective optimization: MOSATS. *Journal of Mathematical Modelling and Algorithms 6,* 2, 213–230.

BAÑOS, R., GIL, C., RECA, J., AND ORTEGA, J. 2010. A pareto-based memetic algorithm for optimization of looped water distribution systems. *Engineering Optimization 42,* 3 (March), 223 – 240.

BENEDETTI, A., FARINA, M., AND GOBBI, M. 2006. Evolutionary multiobjective industrial design: the case of a racing car tire-suspension system. *IEEE Transactions on Evolutionary Computation 10,* 3 (June), 230 – 244.

BEYER, H.-G. AND SCHWEFEL, H.-P. 2002. Evolution strategies –a comprehensive introduction. *Natural Computing: an international journal 1,* 1, 3–52.

BLACKWELL, T. AND BRANKE, J. 2006. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation 10,* 4, 459–472.

BOSMAN, P. A. N. 2005. Learning, anticipation and time-deception in evolutionary online dynamic optimization. In *GECCO '05: Proceedings of the 2005 workshops on Genetic and evolutionary computation*. ACM, New York, NY, USA, 39–47.

BOSMAN, P. A. N. AND POUTRÉ, H. L. 2007. Learning and anticipation in online dynamic optimization with evolutionary algorithms: the stochastic case. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM, New York, NY, USA, 1165–1172.

BRANKE, J. 1999a. Evolutionary algorithms for dynamic optimization problems - a survey. Tech. Rep. 387, Insitute AIFB, University of Karlsruhe. Feb.

BRANKE, J. 1999b. Memory enhanced evolutionary algorithms for changing optimization problems. In *Proceedings of the Congress on Evolutionary Computation CEC '99*. IEEE, 1875–1882.

BRANKE, J. 2001. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, Norwell, MA, USA.

BRANKE, J. 2008. Evolutionary algorithms for dynamic optimization problems (EvoDOP). http://www.aifb.uni-karlsruhe.de/ jbr/EvoDOP/.

BRANKE, J., KAULER, T., SCHMIDT, C., AND SCHMECK, H. 2000. A multi-population approach to dynamic optimization problems. In *In Adaptive Computing in Design and Manufacturing*. Springer, 299–308.

BRANKE, J. AND MATTFELD, D. C. 2005. Anticipation and flexibility in dynamic scheduling. *International Journal of Production Research 43,* 15 (August), 3103–3129.

BRANKE, J., SALIHOĞLU, E., AND ŞIMA UYAR. 2005. Towards an analysis of dynamic environments. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. ACM, New York, NY, USA, 1433–1440.

BRANKE, J. AND SCHMECK, H. 2003. *Advances in evolutionary computing: theory and applications*. Natural Computing Series. Springer-Verlag New York, Inc., New York, NY, USA, Chapter Designing evolutionary algorithms for dynamic optimization problems, 239–262.

BRANKE, J., SCHMECK, H., DEB, K., AND REDDY, M. 2004. Parallelizing multi-objective evolutionary algorithms: cone separation. In *Congress on Evolutionary Computation, 2004*. Vol. 2. 1952–1957.

BUI, L. T. 2007. The role of communication messages and explicit niching in distributed evolutionary multi-objective optimization. Ph.D. thesis, University of New South Wales.

BUI, L. T., ABBASS, H. A., AND ESSAM, D. 2009. Local models–an approach to distributed multi-objective optimization. *Comput. Optim. Appl. 42,* 1, 105–139.

BUI, L. T., ESSAM, D., AND HUSSEIN A. ABBASS. 2010. *Parallel and Distributed Computational Intelligence*. Studies in Computational Intelligence, vol. 269. Springer, Chapter The Role of Explicit Niching and

Communication Messages in Distributed Evolutionary Multi-objective Optimization, 181–206.

BUI, L. T., NGUYEN, M.-H., BRANKE, J., AND ABBASS, H. A. 2007. *Multiobjective Problem Solving from Nature. From Concepts to Applications*. Natural Computing Series. Springer Berlin Heidelberg, Chapter Tackling Dynamic Problems with Multiobjective Evolutionary Algorithms, 77–91.

BÄCK, T. 1998. On the behavior of evolutionary algorithms in dynamic fitness landscapes. In *Proceedings of the Congress on Evolutionary Computation, 1998. CEC 98*. IEEE, 446–451.

CAHON, S. MELAB, N. AND TALBI, E.-G. 2004. Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics 10,* 3, 357 – 380.

CÁMARA, M. 2007. Optimización multiobjetivo paralela en entornos dinámicos. Tech. rep., Dept. de Arquitectura y Tecnología de Computadores, Universidad de Granada. June.

CÁMARA, M. AND ORTEGA, J. 2007. Parallel evolutionary algorithms for dynamic multiobjective optimization. In *HPC-Europa report 2007*, P. Alberigo, G. Erbacci, F. Garofalo, and S. Monfardini, Eds. Vol. 1. CINECA, 334–337.

CÁMARA, M., ORTEGA, J., AND DE TORO, F. 2007a. Parallel processing for multi-objective optimization in dynamic environments. In *Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS '07)*. 1–8.

CÁMARA, M., ORTEGA, J., AND DE TORO, F. 2007b. The parallel single front genetic algorithm (PSFGA) for dynamic multi-objective optimization. In *Proceedings of the 9th International Work-Conference on Artificial Neural Networks (IWANN '07)*, F. Sandoval et al., Eds. Number 4507 in LNCS. Springer-Verlag, 300–307.

CÁMARA, M., ORTEGA, J., AND DE TORO, F. 2008a. A diversity enhanced single front multiobjective algorithm for dynamic optimization problems. In *Proceedings of the 1st International Conference on Metaheuristics and Nature Inspired Computing (META'08)*.

CÁMARA, M., ORTEGA, J., AND DE TORO, F. 2008b. High performance computing for dynamic multi-objective optimisation. *High Performance Systems Architecture 1,* 4, 241–250.

CÁMARA, M., ORTEGA, J., AND DE TORO, F. 2008c. Parallel multi-objective optimization evolutionary algorithms in dynamic environments. In *Proceedings of The First International Workshop on Parallel Architectures and Bioinspired Algorithms*, J. Lanchares, F. Fernández, and J. L. Risco-Martín, Eds. Vol. 1. 13–20.

CÁMARA, M., ORTEGA, J., AND DE TORO, F. 2009a. Medidas de rendimiento para optimización dinámica multiobjetivo. In *Actas del VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'09)*. 357 – 364.

CÁMARA, M., ORTEGA, J., AND DE TORO, F. 2009b. Performance measures for dynamic multi-objective optimization. In *Proceedings of the 10th International Work-Conference on Artificial Neural Networks (IWANN '09)*, J. Cabestany, A. Prieto, and F. Sandoval, Eds. Lecture Notes in Computer Science, vol. 5517. Springer-Verlag, 760 – 767.

CÁMARA, M., ORTEGA, J., AND DE TORO, F. 2010. *Advances in Multi-Objective Nature Inspired Computing*. Studies in Computational Intelligence, vol. 272. Springer, Chapter Approaching Dynamic Multi-objective Optimization Problems by Using Parallel Evolutionary Algorithms, 59–80.

CANTU-PAZ, E. 2000. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA.

CHEN, C., LUO, J., AND PARKER, K. 1998. Image segmentation via adaptive K-mean clustering and knowledge-based morphological operations with biomedical applications. *IEEE Transactions on Image Processing 7,* 12 (December), 1673–1683.

CHO, J. R., JEONG, H. S., AND YOO, W. S. 2002. Multi-objective optimization of tire carcass contours using a systematic aspiration-level adjustment procedure. *Computational Mechanics 29,* 6 (november), 498–509.

COELLO, C. A. 2008. Bibliography on evolutive algorithms for multiobjective optimization. http://www.lania.mx/ ccoello/EMOO/.

COELLO, C. A., LAMONT, G. B., AND VAN VELDHUIZEN, D. A. 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems,* 2nd ed. Genetic and Evolutionary Computation Series. Springer.

CORNE, D. AND KNOWLES, J. 2003. No free lunch and free leftovers theorems for multiobjective optimization problems. In *Proceedings of the Evolutionary Multi-Criterion Optimization EMO '2003*. LNCS, vol. 2632. Springer, 327–341.

CZYZAK, P. AND JASZKIEWICZ, A. 1998. Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis 7,* 1, 34–47.

DARWIN, C. 1995. *The Origin of Species.* Gramercy.

DASGUPTA, D. 1998. *Artficial Immune Systems and Their Applications.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

DE TORO, F., ORTEGA, J., ROS, E., MOTA, S., PAECHTER, B., AND MARTÍN, J. M. 2004. PSFGA: Parallel processing and evolutionary computation for multiobjective optimisation. *Parallel Computing 30,* 721–739.

DE TORO, F., VIDAL, E. R., MOTA, S., AND ORTEGA, J. 2002. PSFGA: A parallel genetic algorithm for multiobjective optimization. *Proceedings of the Euromicro Conference on Parallel, Distributed and Network-Based Processing.*

DEB, K. 2001. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons, Inc., New York, NY, USA.

DEB, K., AGRAWAL, S., PRATAP, A., AND MEYARIVAN, T. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In *PPSN VI: Proceedings of the 6th International Con-*

ference on Parallel Problem Solving from Nature. Number 1917 in LNCS. Springer-Verlag, 849–858.

DEB, K., UDAYA BHASKARA RAO N, AND KARTHIK, S. 2007. Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling. In *Proceedings of the 4th International Conference, EMO 2007*. Number 4403 in LNCS. Springer-Verlag, 803–817.

DEB, K., ZOPE, P., AND JAIN, A. 2003. Distributed computing of pareto-optimal solutions using multi-objective evolutionary algorithms. In *Proceedings of the Second Evolutionary Multi-Criterion Optimization (EMO-03) Conference*. LNCS, vol. 2632. 535–549.

DEMPSEY, I., O'NEILL, M., AND BRABAZON, A. 2009. *Foundations in Grammatical Evolution for Dynamic Environments*. Studies in Computational Intelligence, vol. 194. Springer.

DEMŠAR, J. 2006. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res. 7*, 1–30.

DORIGO, M. AND STÜTZLE, T. 2004. *Ant Colony Optimization*. Bradford Books. MIT Press.

DOWSLAND, K. A. 1993. *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc., New York, NY, USA, Chapter Simulated annealing, 20–69.

DOXIADIS, A. AND PAPADIMITRIOU, C. 2009. *Logicomix: An Epic Search for Truth*. Bloomsbury.

FARINA, M., DEB, K., AND AMATO, P. 2004. Dynamic multiobjective optimization problems: Test cases, approximations, and applications. *IEEE Trans. on Evolutionary Computation 8*, 425–442.

FENG, W., BRUNE, T., CHAN, L., CHOWDHURY, M., KUEK, C., AND LI, Y. 1997. Benchmarks for testing evolutionary algorithms. Tech. Rep. CSC-97006, Faculty of Engineering, University of Glasgow.

FOGEL, L. J. 1999. *Intelligence through simulated evolution: forty years of evolutionary programming*. John Wiley & Sons, Inc., New York, NY, USA.

FONSECA, C. M. AND FLEMING, P. J. 1993. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*. 416–423.

FONSECA, C. M. AND FLEMING, P. J. 1996. On the performance assessment and comparison of stochastic multiobjective optimizers. In *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*. Springer-Verlag, London, UK, 584–593.

GARCÍA, S. AND HERRERA, F. 2008. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research 9*, 2677–2694.

GASPAR, A. 1999. From GAs to artificial immune systems: improving adaptation in time dependent optimization. In *In Proceedings of the Congress on Evolutionary Computation*. IEEE Press, 1859–1866.

GLOVER, F. 1989. Tabu Search — Part I. *ORSA Journal on Computing 1,* 3, 190–206.

GLOVER, F. AND LAGUNA, M. 2000. Fundamentals of scatter search and path relinking. *Control and Cybernetics 29,* 3, 653–684.

GLOVER, F. W. AND KOCHENBERGER, G. A., Eds. 2003. *Handbook of Metaheuristics*. International Series in Operations Research & Management Science, vol. 57. Springer.

GOH, C.-K. AND TAN, K. C. 2009. *Evolutionary Multi-objective Optimization in Uncertain Environments*. Studies in Computational Intelligence, vol. 186. Springer.

GOLDBERG, D. E. AND SMITH, R. E. 1987. Nonstationary function optimization using genetic algorithm with dominance and diploidy. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 59–68.

GRAHAM, P. 1993. *On Lisp*. Prentice Hall. Freely available in http://www.paulgraham.com/onlisp.html.

GREFENSTETTE, J. J. 1999. Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In *Proceedings of the Congress on Evolutionary Computation, 1999. CEC 99*. Vol. 3. IEEE, 2031–2038.

GUPTA, A. AND SIVAKUMAR, A. 2006. Pareto control in multi-objective dynamic scheduling of a stepper machine in semiconductor wafer fabrication. In *Proceedings of the 2006 Winter Simulation Conference*. IEEE Computer Society, Los Alamitos, CA, USA, 1749–1756.

HADAD, B. S. AND EICK, C. F. 1997. Supporting polyploidy in genetic algorithms using dominance vectors. In *EP '97: Proceedings of the 6th International Conference on Evolutionary Programming VI*. Springer-Verlag, London, UK, 223–234.

HATZAKIS, I. AND WALLACE, D. 2006. Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, New York, NY, USA, 1201–1208.

HORN, J. AND NAFPLIOTIS, N. 1993. Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Tech. Rep. IlliGAl Report 93005, Urbana, Illinois, USA.

JIN, Y. AND BRANKE, J. 2005. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation 9,* 3, 303–317.

KATZ, V. J. 1998. *A History of Mathematics: An Introduction*, 2nd ed. Addison Wesley.

KEIJZER, M., MERELO, J. J., ROMERO, G., AND SCHOENAUER, M. 2002. Evolving objects: A general purpose evolutionary computation library. In *Selected Papers from the 5th European Conference on Artificial Evolution*. Springer-Verlag, London, UK, 231–244.

KENNEDY, J. AND EBERHART, R. C. 1995. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks IV*. Vol. 4. 1942 – 1948.

KNOWLES, J. AND CORNE, D. 1999. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Proceedings of the Congress on Evolutionary Computation*, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, Eds. Vol. 1. IEEE Press, Mayflower Hotel, Washington D.C., USA, 98–105.

KNOWLES, J. AND CORNE, D. 2002. On metrics for comparing nondominated sets.

KNOWLES, J., THIELE, L., AND ZITZLER, E. 2006. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich. Feb.

KOISHI, M. AND SHIDA, Z. 2006. Multi-objective design problem of tire wear and visualization of its pareto solutions. *Tire Science and Technology 34,* 3 (September), 170–194.

KÖPPEN, M. 2004. No-free-lunch theorems and the diversity of algorithms. In *Proceedings of the Congress on Evolutionary Computation CEC '2004*. Vol. 1. 235 – 241.

KÖVESI, B., BOUCHER, J.-M., AND SAOUDI, S. 2001. Stochastic K-means algorithm for vector quantization. *Pattern Recognition Letters 22,* 6-7, 603–610.

KOZA, J. R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

LAUMANNS, M. 2001. SPEA2 implementation for PISA. http://www.tik.ethz.ch/ sop/pisa/. ETH Zurich.

LAUMANNS, M. AND LAUMANNS, N. 2005. Evolutionary multiobjective design in automotive development. *Applied Intelligence 23,* 1, 55–70.

LEE, L. H., TENG, S., CHEW, E. P., KARIMI, I. A., LYE, K. W., LENDERMANN, P., CHEN, Y., AND KOH, C. H. 2005. Application of multi-objective simulation-optimization techniques to inventory management problems. In *WSC '05: Proceedings of the 37th conference on Winter simulation*. Winter Simulation Conference, 1684–1691.

LI, X., BRANKE, J., AND KIRLEY, M. 2007. On performance metrics and particle swarm methods for dynamic multiobjective optimization problems. In *IEEE Congress on Evolutionary Computation*. 576–583.

LUKE, S. 2009. *Essentials of Metaheuristics*. available at http://cs.gmu.edu/∼sean/book/metaheuristics/.

LUNA, F., NEBRO, A. J., AND ALBA, E. 2006. Parallel evolutionary multiobjective optimization. See Nedjah et al. (2006), 33–56.

LUNA, F., NEBRO, A. J., ALBA, E., AND DURILLO, J. J. 2008. Solving large-scale real-world telecommunication problems using a grid-based genetic algorithm. *Engineering Optimization 40,* 11, 1067 – 1084.

MEHNEN, J., WAGNER, T., AND RUDOLPH, G. 2006. Evolutionary optimization of dynamic multiobjective functions. Interner Bericht des Sonderforschungsbereichs 531 *Computational Intelligence* CI–204/06, Universität Dortmund. April.

MITCHELL, M. 1996. *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, USA.

MORI, N., KITA, H., AND NISHIKAWA, Y. 1996. Adaption to a changing environment by means of the thermodynamical genetic algorithm. In *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*. Springer-Verlag, London, UK, 513–522.

MORI, N., KITA, H., AND NISHIKAWA, Y. 1998. Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*. Springer-Verlag, London, UK, 149–158.

MORRISON, R. 2003. Performance measurement in dynamic environments. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, J. Branke, Ed. 5–8.

NAVARRO, A. AND ALLEN, C. 1997. Adaptive classifier based on K-means clustering and dynamic programming. In *Document Recognition IV*. Vol. 36. 31–38.

NEDJAH, N., DE MACEDO MOURELLE, L., AND ALBA, E., Eds. 2006. *Parallel Evolutionary Computations*. Studies in Computational Intelligence, vol. 22. Springer.

O'NEILL, M. AND RYAN, C. 2003. *Grammatical Evolution*. Genetic Programming, vol. 4. Springer.

ROSSI, C., ABDERRAHIM, M., AND DÍAZ, J. C. 2008. Tracking moving optima using Kalman-based predictions. *Evolutionary Computation 16,* 1, 1–30.

ROWE, J. E., VOSE, M. D., AND WRIGHT, A. H. 2009. Reinterpreting no free lunch. *Evolutionary Computation 17,* 1, 117–129.

SALOMON, R. AND EGGENBERGER, P. 1998. Adaptation on the evolutionary time scale: A working hypothesis and basic experiments. In *AE '97: Selected Papers from the Third European Conference on Artificial Evolution*. Springer-Verlag, London, UK, 251–262.

SCHAFFER, J. 1985. Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*. 93–100.

SCHAFFER, J. D. 1984. Multiple objective optimization with vector evaluated genetic algorithms. Ph.D. thesis, Vanderbilt University.

SCHEFFÉ, H. 1999. *The Analysis of Variance*. Wiley-Interscience.

SEIBEL, P. 2004. *Practical Common Lisp*. Apress.

SIERPINSKI, W. 2000. *General topology*. Dover. Originally published: Toronto, Canada : University of Toronto Press, 1956.

SILVERMAN, B. W. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC.

SRINIVAS, N. AND DEB, K. 1994. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation 2,* 221–248.

STORN, R. AND PRICE, K. 1997. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization 11,* 4, 341–359.

STREICHERT, F., ULMER, H., AND ZELL, A. 2005. Parallelization of multi-objective evolutionary algorithms using clustering algorithms. In *Proceedings of the Third International Conference, EMO 2005*. Lecture Notes in Computer Science, vol. 3410. Springer Berlin, 92–107.

TALBI, E.-G. 2006. *Parallel Combinatorial Optimization*. Wiley.

TALBI, E.-G. 2009. *Metaheuristics: From Design to Implementation*. Wiley.

TINÓS, R. AND YANG, S. 2007. A self-organizing random immigrants genetic algorithm for dynamic optimization problems. *Genetic Programming and Evolvable Machines 8,* 3, 255–286.

TOMASSINI, M. 1999. Parallel and distributed evolutionary algorithms: A review. Tech. rep., University of Lausanne.

TROJANOWSKI, K. AND MICHALEWICZ, Z. 1999. Searching for optima in non-stationary environments. In *Proceedings of the Congress on Evolutionary Computation CEC '99*. IEEE Press, Piscataway, NJ, 1843–1850.

VAN VELDHUIZEN, D. A. 1999. Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations. Ph.D. thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH.

VAN VELDHUIZEN, D. A. AND LAMONT, G. B. 2000. On measuring multiobjective evolutionary algorithm performance. In *Proceedings of the Congress on Evolutionary Computation '2000*. press, 204–211.

VAN VELDHUIZEN, D. A., ZYDALLIS, J. B., AND LAMONT, G. B. 2003. Considerations in engineering parallel multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation 7,* 144– 173.

VAVAK, F., FOGARTY, T. C., AND JUKES, K. 1996. A genetic algorithm with variable range of local search for tracking changing environments. In *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*. Springer-Verlag, London, UK, 376–385.

WANG, H., WANG, D., AND YANG, S. 2009. A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. *Soft Computing 13,* 8-9, 763–780.

WEICKER, K. 2002. Performance measures for dynamic environments. In *PPSN VII: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*. Springer-Verlag, London, UK, 64–76.

WEICKER, K. AND WEICKER, N. 1999. On evolution strategy optimization in dynamic environments. In *Congress on Evolutionary Computation '1999*. IEEE Press, 2039–2046.

WOLPERT, D. H. AND MACREADY, W. G. 1995. No free lunch theorems for search. Tech. Rep. SFI-TR-95-02-010, Santa Fe Institute.

WOLPERT, D. H. AND MACREADY, W. G. 1997. No free lunch theorems for optimization. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION 1,* 1, 67–82.

WOLPERT, D. H. AND MACREADY, W. G. 2005. Coevolutionary free lunches. *IEEE Transactions on Evolutionary Computation 9,* 6, 721–735.

YANG, S. 2008. Genetic algorithms with memory-and elitism-based immigrants in dynamic environments. *Evolutionary Computation 16,* 3, 385–416.

YANG, S., ONG, Y.-S., AND JIN, Y. 2007. *Evolutionary Computation in Dynamic and Uncertain Environments (Studies in Computational Intelligence).* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

YU, X., TANG, K., CHEN, T., AND YAO, X. 2009. Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization. *Memetic Computing 1,* 1 (March), 3–24.

ZITZLER, E., DEB, K., AND THIELE, L. 2000. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation 8,* 2, 173–195.

ZITZLER, E., LAUMANNS, M., AND THIELE, L. 2001. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. TIK Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland.

ZITZLER, E., LAUMANNS, M., AND THIELE, L. 2002. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, K. Giannakoglou et al., Eds. International Center for Numerical Methods in Engineering (CIMNE), 95–100.

ZITZLER, E., LAUMANNS, M., THIELE, L., FONSECA, C. M., AND DA FONSECA, V. G. 2002. Why quality assessment of multiobjective optimizers is difficult. In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 666–674.

ZITZLER, E., THIELE, L., LAUMANNS, M., FONSECA, C. M., AND GRUNERT DA FONSECA, V. 2003. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation 7*, 2, 117–132.

Glossary

Boxplot

A way of graphically depicting groups of numerical data through their five-number summaries: the smallest observation (sample minimum), lower quartile (Q1), median (Q2), upper quartile (Q3), and largest observation (sample maximum). A boxplot may also indicate which observations, if any, might be considered outliers

Crowding distance

A parameter used by some MOEA to promote the diversity within the population. It refers to a minimum distance that should be kept among all the solutions in the population.

Decision space

It is the range of the variables that an optimization problem function can take or the values that we can change when looking for a better solution.

Dynamic Optimization Problem (DMO)

A problem where the restrictions or the objective functions change with time.

Fitness function

The function that evaluates the possible values that an optimization problem can take in order to choose the best from them.

Front or Rank

A *front* or *rank* is a subset of the population where all the contained solutions are mutually non-dominated.

Measure

A function that assigns a real value to a given input set.

Metaheuristic

A heuristic method used to solve a problem by using other heuristic methods.

Metrics

Mathematical concept for distances.

Multi-evaluated problem

A kind of problem where the decision variables are in a space vector but the objective space is $\mathbb{R}$.

Multi-objective optimization problem

A problem where there are more than one decision and objective variables to optimize.

Non-dominated sorting

A procedure in which all the multi-objective solutions from a population are classified into different fronts or ranks. Every front *i-th* contain all the solutions that are dominated by at least one solution from the fronts before i but that dominate all the solutions in the fronts after i.

Objective space

It is the domain of the values that an optimization problem function can take or the values that we are looking to optimize for the fitness function.

Optimization problem

A function $f(x)$ in $\mathbb{R}$ that represents a process where at least one value of $f(x)$ is less or equal to the rest of values of $f(x)$. There can be maximization optimization problems, where at least one value of $f(x)$ is above or equal to the rest of values.

Pareto dominance

The (strict) Pareto dominance is the binary relation that represents a strict partial order of the objective space of an optimization problem.

Pareto front approximation

The set of mutually incomparable objective vectors.

Pareto optimal

The *decision* or *objective* vector that is not dominated by any other vector in the *objective* space.

Pareto set approximation

The set of mutually incomparable solutions.

Performance measure

A specific *measure* that assigns a real value to a given input set of solutions for a given problem. The output value evaluates the set of the solutions for the problem.

Post-hoc test

A test used a posteriori on a group of data after the null hypothesis has been rejected. A post-hoc test is expected to tell which data groups significantly differ from the other ones.

Search space

See *Decision space*.

Set of solutions

The set of solutions for an optimization problem is the set of values for the decision variables that make the objective values not worse than any other objective value.

Weak Pareto dominance

The weak Pareto dominance is the binary relation that represents a partial order of the objective space of an optimization problem.

Index