UNIVERSIDADE DE VIGO

DEPARTAMENTO DE ENXEÑERÍA QUÍMICA

NEW HEURISTICS FOR GLOBAL OPTIMIZATION OF COMPLEX BIOPROCESSES

Memoria para optar al grado de Doctor Europeus por la Universidad de Vigo presentada por

José Alberto Egea Larrosa

Vigo, 2008

Autorización

Los Doctores **Julio Rodríguez Banga**, Investigador Científico del Instituto de Investigaciones Marinas de Vigo (C.S.I.C.), y **Rafael Martí Cunquero**, Catedrático de Universidad de la Universitat de València

CERTIFICAN:

Que la memoria adjunta, titulada "New Heuristics for global optimization of complex bioprocesses", que para optar al grado de Doctor presenta D. José Alberto Egea Larrosa, ha sido realizada bajo su inmediata dirección en el Instituto de Investigaciones Marinas del C.S.I.C. y, considerando que constituye trabajo de Tesis, autorizan su presentación en la Universidad de Vigo.

Vigo, 9 de Enero de 2008

Fdo.: Dr. Julio Rodríguez Banga

Fdo.: Dr. Rafael Martí Cunquero

Agradecimientos

Quiero dedicar unas líneas de agradecimientos a las personas que, de una u otra forma, han tenido alguna influencia en el desarrollo de este trabajo.

Comienzo con mis directores de tesis, los doctores Julio Rodríguez Banga y Rafael Martí Cunquero, en primer lugar por lo mucho que he aprendido a su lado durante este tiempo, y en segundo (y no por ello menos importante), por el trato que me han dispensado. Mención especial debo hacer a Julio, con quien más he convivido. Debo agradecerle la paciencia que ha tenido conmigo en muchas ocasiones (no sé si alguna vez volverá a tener un doctorando tan "desobediente" como yo) y, sobre todo, la sensación de confianza en mí y en mi trabajo que me ha transmitido, especialmente durante la última etapa. Esto es algo muy importante y que aprecio mucho. Además, el ambiente de trabajo ha sido inmejorable, reinando casi siempre el buen humor. No me olvido de Rafa, con el que, debido a la distancia geográfica, he convivido menos pero con el que me he mantenido en contacto permanente. En todo momento me ha tratado como a un amigo y me ha escuchado con atención hasta el final cuando tenía algo que sugerir o aportar.

No me hubiera atrevido a escribir la tesis íntegramente en inglés de no haber sabido que tendría el apoyo y colaboración de mis jefes y la ayuda de un profesional. Gracias, Manuel, por tus correcciones.

Durante cuatro años he disfrutado de financiación económica proporcionada por el Programa de Formación del Profesorado Universitario (PFPU) del Ministerio de Educación y Ciencia. Dentro de este mismo programa, he disfrutado de ayudas económicas para realizar estancias breves en el extranjero. Con respecto a éstas, quiero agradecer al Profesor Kenneth Holmström, de Mälardalens University (Suecia), su acogida durante casi tres meses a finales de 2005, y a Nils-Hassan Quttineh (Nisse) por su ayuda respecto al trabajo allí realizado. A finales de 2006 realicé una estancia breve en Supélec (Francia), donde gracias al Dr. Emmanuel Vazquez aprendí lo que es el kriging y cómo aplicarlo a optimización global. Guardo un excelente recuerdo de esa estancia tanto por la forma de trabajar de Emmanuel como por el tiempo y el interés que me dedicó. Agradezco también a Julien Villemonteix las conversaciones e intercambios de ideas que me ayudaron a seguir aprendiendo durante esa estancia y posteriormente.

El trabajo de tesis no es sólo trabajo en cuanto a que el entorno tiene, a mi juicio, no poca influencia sobre el ánimo y las ganas que se ponen. En este sentido, he disfrutado de un excepcional ambiente de trabajo y amistad con la gente del Grupo de Ingeniería de Procesos del I.I.M. Si tuviera que comenzar de nuevo, elegiría sin duda al mismo grupo de compañeros: Carlos, Irene, Míriam, Marcos, Antonio, Luis, María, Óscar, Sonia, Eva, Oliver, Martin, Amaya. A todos tengo que agradecer las sugerencias y/o preguntas sobre SSm que me han ayudado a encontrar errores, a mejorarlo y a aprender un poco más. Quiero resaltar a algunos por su relación más o menos directa con mi trabajo. Oliver y Martin: gracias por las largas conversaciones e intercambios de ideas sobre los algoritmos que estábamos implementando.

Sonia y Eva: gracias por vuestra ayuda y aclaraciones en la parte de optimización dinámica. Óscar: gracias por contestar a tantas y tantas preguntas sobre Matlab, optimización, etc..., sin decirme ni una sola vez que no, ni siquiera que esperara un momento. María, gracias por las aclaraciones sobre estimación de parámetros, diseño óptimo, identificabilidad, etc, pero sobre todo gracias por las conversaciones sobre nuestras vidas; esa oficina no habría sido lo mismo sin ti, y en general mi vida en el I.I.M. no habría sido lo mismo sin ti. Gracias por ser una amiga tan especial, por comprenderme tan bien y por hacer que te comprenda tan bien. Y a ti, Breva, ¿qué te voy a decir? La fatalidad hizo que hace un año y pico tu vida diese un giro y una de las consecuencias fue que yo tuve que empezar a disfrutar menos tiempo de ti. Sin embargo, para ese entonces yo ya había ganado un hermano y habíamos llegado a esa fase en la que no importa el tiempo que pase ni la distancia a la que se esté para que alguien haya quedado incrustado en el corazón para siempre. Como escribí antes a María, la vida no habría sido igual sin ti. De hecho empezó a ser distinta una vez que ya no estabas para que te estafara 15 minutos cada mañana para ir al trabajo o para que te lanzara la zapatilla cuando te me quedabas sopa en el sofá por la noche. Gracias por ser así y haberme enseñado tanto.

Mis padres, a quienes dedico este trabajo con toda mi ilusión, han sido y son mi apoyo constante, mi refugio y el espejo en el que mirarme. Os he tenido en mi cabeza en todo momento y vuestro pensamiento ha acompañado a todos y cada uno de los pasos dados hacia delante durante estos años. A ambos, muchas gracias por vuestro amor, por cultivar mis inquietudes y por vuestra confianza ciega en mí. A vosotros os lo debo todo. Os quiero.

Creo firmemente que el estado de ánimo influye de forma decisiva en el resultado de las acciones y/o trabajos que realizamos. En este sentido, quiero agradecer a la gente que ha contribuido a que mi vida en Vigo haya sido feliz y completa. A Susi, Clemente y Laura, amigos para toda la vida con los que he pasado momentos inolvidables. A mis compañeros del equipo de basket que, aunque creo que ni siquiera saben a lo que me dedico, me llamaron a París para jugar con ellos a mi vuelta y me "renovaron" automáticamente cuando tuve la lesión. A mis compañeros de piso post-Luis, primero Vicky y luego Raúl, que me hicieron sentir triste cuando sabía que se iban de casa pero que se han quedado como buenos amigos enriqueciendo un poco más mi vida. Mis amores en Murcia (o más correctamente, de origen murciano esparcidos por la península) han sido muy importantes para mí a pesar de no vivir el día a día con ellos. Con Manolo, Palbe, Jesús y Lauricia me siento como en mi hogar por el simple hecho de verlos y estar juntos. Con nadie me siento tan a gusto y con nadie me río tanto. Seremos muy tontos, pero qué bien lo pasamos, ¿verdad?

Muchos jóvenes doctores con los que he hablado coincidían en decir que la última parte de la tesis, la escritura, los últimos experimentos, las correcciones, etc..., es un período muy largo y de mucho estrés. Para mí, sin embargo, ha venido a coincidir con una de las etapas más felices de mi vida (quizás la más feliz). Varios han sido los motivos para que esto sea así, y el más importante de ellos eres tú, Rebeca.

A mis padres

Oigo y olvido,
veo y recuerdo,
hago y aprendo

Proverbio chino

Summary

Optimization problems arising from the biotechnological and food industries are usually of non-convex nature and they often exhibit several local minima. Even though advances in global optimization research have been outstanding in recent years, the current state-of-the-art is not completely satisfactory, specially when one considers the global optimization of complex process models (typical of biotechnological and food industries). These models are complex due to their dynamic behavior and large number of states. Besides, one of the most important drawbacks for optimizing these complex models is the computation time required to perform every simulation. Due to the large number of differential and algebraic equations (DAE's) defining the mathematical models which describe complex processes and/or full industrial plants, the time needed to perform a single simulation may vary between some minutes and hours on a standard personal computer. This can lead to unaffordable computation times from the practical point of view when the optimization of such processes is carried out.

The reasons exposed above advise to treat complex models as black boxes in many situations, that is, as a simple relationship between inputs and outputs without further information about relationships among the decision variables. For this kind of problems, stochastic global optimization methods (and metaheuristics in particular) have proved to be efficient and robust. Indeed, even though these methods can not ensure the convergence to the global optimum, they provide very good solutions in practice (the global optimum in some cases) in reasonable computation times. Besides, stochastic methods permit to treat mathematical models as black-boxes and are easy to implement, making them robust for any kind of problem.

The use of the so-called metamodels allows to build surrogate models which interpolate or approximate the original models, and predict their function values with a certain probability, being less difficult to evaluate from the computational point of view. Taking advantage of

their statistical properties, these surrogate models allow us to formulate hypotheses about the location of the global optimum and to find it (or high quality solutions) in a number of simulations much lower than these employed by traditional optimization methods, thus considerably reducing the final optimization time.

In the first part of this work we present an introduction to global optimization in the biotechnological area, including the main type of existing problems and the available optimization methods to solve them. An introduction to a special class of stochastic methods (metaheuristics) is provided, pointing out the most popular and successful among them. Considering that our proposed method is based on the scatter search methodology, we describe it in Chapter 3.

In the second part the methodology proposed for the optimization of complex bioprocesses is explained. We present a scatter search-based algorithm for the global optimization of nonlinear dynamic systems. A set of new heuristics and improved features have been developed to handle the main drawbacks inherent to this kind of problems. We have also developed another optimization algorithm (based on scatter search too) which makes use of surrogate models for the optimization of computationally expensive problems. In particular, the algorithm uses a kriging interpolation algorithm which provides predictions and statistics associated to those predictions, in order to minimize the number of simulations to locate the global optimum. The scatter search framework makes the algorithm autonomous to select the set of points in which the predictions must be done. The associated software tools for both algorithms have been developed, and we present their documentation in Appendix A. Their effectiveness is demonstrated by the resolution of a set of benchmark problems.

The final part of this work is dedicated to the application of the proposed methodologies to different problems arising in the biotechnological and food industries. The three main types of problems described in the first part are considered, and the performances of our algorithms are compared with those of other state-of-the-art optimization algorithms, showing that our approach is efficient and robust for the global optimization of this kind of problems.

Resumen

Los problemas de optimización que surgen en el campo de los procesos biotecnológicos y alimentarios suelen tener una naturaleza no convexa, existiendo con frecuencia numerosos óptimos locales. Aunque los avances en optimización global han sido notables en los últimos años, el estado actual no es del todo satisfactorio, sobre todo cuando se considera la optimización global de modelos de procesos complejos (típicos de las industrias biotecnológicas y alimentarias). Estos modelos son complejos debido a su comportamiento dinámico y al elevado número de estados. Además, uno de los problemas más importantes para la optimización de estos modelos complejos es el tiempo de computación necesario para llevar a cabo cada simulación. Debido al elevado número de ecuaciones diferenciales y algebraicas existentes en los modelos que describen procesos complejos o plantas completas, el tiempo necesario para realizar una única simulación puede ser del orden de varios minutos o incluso horas en un ordenador convencional. Esto puede conducir a tiempos de computación inabordables desde el punto de vista práctico cuando se lleva a cabo la optimización de dichos procesos.

Las razones anteriores aconsejan, en muchas ocasiones, tratar los modelos complejos como cajas negras, es decir, como una relación simple entre entradas y salidas sin que se tenga información sobre la relación entre las variables. Para este tipo de problemas, los métodos de optimización global (y las metaheurísticas en particular) han demostrado su eficiencia y robustez. En efecto, aunque estos métodos no aseguran la convergencia al óptimo global, en la práctica proporcionan buenas soluciones (el óptimo global en muchos casos) en tiempos de computación razonables. Además, estos métodos permiten tratar los modelos matemáticos como cajas negras y son de fácil implementación, lo que les proporciona robustez y los hace útiles para cualquier tipo de problema.

El uso de los llamados metamodelos permite construir modelos sustitutos que interpolan o aproximan los modelos originales y predicen sus valores con cierta probabilidad, siendo mucho menos difíciles de evaluar desde el punto de vista computacional. Aprovechando

sus propiedades estadísticas, estos modelos sustitutos permiten realizar hipótesis sobre la localización del óptimo global y llegar a él (o a soluciones de alta calidad) en un número de simulaciones mucho menor que los métodos de optimización tradicionales, y por tanto reduciendo considerablemente el tiempo final de optimización.

En la primera parte de este trabajo se presenta una introducción a la optimización global en el área biotecnológica, incluyendo los principales tipos de problemas existentes y los métodos de optimización disponibles para resolverlos. También se hace una introducción específica a una clase de métodos estocásticos (las metaheurísticas), destacando las más populares y exitosas de entre ellas. Considerando que el método de optimización propuesto en esta tesis está basado en la metodología conocida como *scatter search* (búsqueda dispersa en castellano), ésta se describe en el Capítulo 3.

En la segunda parte se presenta la metodología propuesta para la optimización de bioprocesos complejos. Se presenta un algoritmo de optimización global basado en *scatter search* para la optimización de sistemas dinámicos no lineales. Se han desarrollado un conjunto de nuevas heurísticas y características mejoradas para intentar resolver los principales inconvenientes asociados a este tipo de problemas. Se ha desarrollado un segundo algoritmo de optimización global (también basado en *scatter search*) que hace uso de modelos sustitutos para la optimización de problemas computacionalmente costosos. En concreto, el algoritmo usa una interpolación basada en *kriging* que proporciona predicciones y estadísticas asociadas a ellas para minimizar el número de simulaciones necesarias para localizar el óptimo global. El hecho de estar basado en *scatter search* hace que el algoritmo elija automáticamente el conjunto de puntos sobre los que se hará la predicción. Las herramientas de software asociadas a ambos algoritmos se documentan en el Apéndice A. Su efectividad queda demostrada mediante la resolución de una serie de problemas como banco de pruebas.

La parte final de este trabajo se dedica a la aplicación de las metodologías propuestas a diferentes problemas de las industrias biotecnológicas y alimentarias. Se consideran los tipos de problemas descritos en la primera parte. El comportamiento de nuestros algoritmos se compara con el de otros algoritmos de optimización global que constituyen el estado actual, demostrando que las metodologías propuestas son eficientes y robustas para cumplir con el objetivo propuesto.

Objectives

The main objective of this work consists in developing a methodology for the global optimization of complex bioprocesses (i.e., processes from the biotechnological and food industries). Mathematical models describing such processes are often non-linear and multimodal. Thus, their optimization is a difficult and time-consuming task where the state-of-the-art optimization algorithms often fail. To successfully comply with this main objective, a set of sub-objectives has been formulated:

- Review of the type of problems arising in the bioprocess industry optimization and the type of optimization algorithms available to solve such problems.
- Review of the most promising metaheuristics and their application to (bio)process engineering optimization problems.
- Analysis of the scatter search and kriging methodologies and their application to the class of problems of our interest.
- Development of heuristics and advanced features to overcome typical drawbacks of some optimization problems which prevent classical optimization methods from solving them.
- Exploitation of the statistical information provided by the kriging interpolation technique to develop patterns of search in the optimization of computationally expensive models.
- Development of software tools to test the proposed methodologies with benchmark and real problems.
- Application of the proposed methodologies to a set of industrial problems covering the most relevant types found in the bioprocess industries.

Objetivos

El principal objetivo de este trabajo consiste en desarrollar una metodología para la optimización global de bioprocesos complejos (procesos de las industrias biotecnológicas y alimentarias). Los modelos matemáticos que describen dichos procesos suelen ser no lineales y multimodales. Por tanto, su optimización es una tarea compleja y costosa (en términos de tiempo de computación) en la que los métodos de optimización global que constituyen el estado actual pueden fallar. Para cumplir con este objetivo principal se han formulado una serie de sub-objetivos:

- Revisión de los tipos de problemas que surgen en la optimización de bioprocesos y de los algoritmos disponibles para su resolución.
- Revisión de las metaheurísticas más prometedoras y de su aplicación a problemas de optimización en ingeniería de (bio)procesos.
- Análisis de las metodologías de *scatter search* y *kriging* y su aplicación a la clase de problemas que nos interesa.
- Desarrollo de heurísticas y características avanzadas para superar los inconvenientes que surgen en cierto tipo de problemas de optimización y que evitan que los métodos clásicos de optimizacion puedan resolverlos.
- Uso de la información estadística proporcionada por la interpolación por *kriging* para desarrollar patrones de búsqueda en la optimización de modelos computacionalmente costosos.
- Desarrollo de herramientas de software para testar las metodologías propuestas en problemas reales.
- Aplicación de las metodologías propuestas en un conjunto de problemas industriales que abarquen los tipos de problemas más relevantes en las industrias de bioprocesos.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Bioprocess Engineering Optimization

Biotechnology is the application of science and engineering to the use of living organisms or substances derived from them, to generate products or to perform functions that can benefit the human condition. It is based on scientific areas such as physiology, molecular biology and molecular genetics as well as on other arising disciplines like genomics, proteomics, transcriptomics and metabolomics. Biotechnology covers a broad segment of science and its industrial applications. It has had a major impact in our society over the last decades due to its quick development and its applications to improve the quality of life.

Bioprocess engineering is the subdiscipline within biotechnology that is responsible for translating the discoveries of science into practical products, processes or systems that can serve the needs of society. Quoting Louis Pasteur, bioprocess engineering is to biotechnology *as the fruit is to the tree.* Neither can exist without the other. Bioprocess engineering enables the development of new products, services and industrial processes in several sectors. Table 1.1 presents a list of biotechnological products in different economic areas.

Bioprocess engineering covers all engineering aspects of biotechnological production. It was developed out of chemical engineering in the mid-20th century, when the initiation of the first antibiotic production on the industrial scale imposed the use of techniques unknown to the engineering profession, like sterility and sterile techniques of manipulating great volumes, microbiological fluid filtration, kinetics of microbial processes, mixing and aeration. The main objectives for the biochemical engineer must encompass the design of an optimal process that:

Activity	Product
Industrial (bio)chemical products	Ethanol, acetone, organic acids, amino acids, biopolymers, enzymes.
Pharmaceutics	Therapeutic proteins, antibodies, vaccines, antibiotics, signal molecules, diagnostic agents, enzyme inhibitors.
Energetics	Biofuels, biogas, hydrogen.
Food	Fermented foods and beverages, probiotics, proteins, additives, amino acids, sugars.
Agriculture	Silage, compost, biological fertilizers, insecticides and pesticides.
Services	Waste water treatments, solid waste processing, analytical agents, tissue and organ engineering.
Mining	Extraction and concentration of metals, secondary oil processing.

Table 1.1: Biotechnological products in different economic areas

(i) provides the desired quality of a final product; (ii) minimizes the total process time and cost through improved operational efficiency; and (iii) falls within the constraints of acceptable market entry (Najafpour, 2006).

It is quite usual to find the term *biochemical engineering* meaning bioprocess engineering. However, there is a difference between them. Bioprocess engineering includes mechanical, electrical and industrial engineers working to apply the principles of their disciplines to biotechnological processes. Some problems such as equipment design, sensor development, control and process optimization can utilize principles from these disciplines (Shuler and Kargi, 1992).

Like in many other scientific and engineering fields, mathematical modeling, optimization and control have become fundamental tools for optimally designing and operating production facilities in the biotechnological process industries (e.g., see Shimizu 1996; Bailey 1998; Steffens et al. 2000; Barton et al. 2000; Banga et al. 2003b,c; Biegler and Grossmann 2004; Floudas et al. 2005)

Since many biotechnological processes are operated in batch or semi-continuous modes, they have an inherently dynamic nature. In this context, there are at least three relevant types of optimization problems (as we will introduce in Section 1.1): optimal operation (dynamic optimization), integrated process design and control, and parameter estimation. These problems can be stated as, or transformed into, nonlinear programming problems subject to dynamic (usually differential algebraic) constraints. Their highly constrained, non-linear and sometimes non-smooth nature often causes non-convexity, and therefore global

optimization methods are needed in order to find suitable solutions.

1.1 Types of optimization problems in bioprocess engineering

In the context of bioprocess engineering optimization there are three types of problems specially relevant (Banga et al., 2003b):

- **Dynamic optimization** (or open-loop optimal control): Given a pre-defined performance index such as profitability, product quality or productivity (often subject to safety or environmental specifications), the aim is to find the optimal operating conditions over a time interval.

- **Integrated design and control:** The aim is to simultaneously find the static variables of the process design, the operating conditions and the controllers' parameters which optimize a combined measure of the plant economics and its controllability, subject to a set of constraints which ensure appropriate dynamic behavior and process specifications.

- **Parameter estimation** (also called the inverse problem): The aim is to find the set of parameters of a mathematical model to obtain the best possible fit to the existing experimental data.

In this study we will consider all these optimization problems as nonlinear programming problems subject to differential-algebraic constraints. The resolution of the differential-algebraic constraints is usually a hard problem. Thus, an approximate method (typically a Runge-Kutta, BDF method, or a similar numerical process) is applied to simultaneously solve the set of DAE's and obtain the state values corresponding to a set of values for the decision variables. Therefore, this kind of complex problem is solved with a black-box sequential method in which the optimization takes place in the set of the decision variables. Given a set of values for the decision variables, the approximate solver of the differential-algebraic constraints computes the associated values for the states. We then test the feasibility of the solution with the set of additional inequality and/or equality constraints functions (if they exist). To sum it up, a remarkable computational effort is associated with the evaluation and the feasibility test of one solution.

1.1.1 Dynamic optimization

For many industrial processes (and specially bioprocesses), computing the optimal operation policies becomes fundamental to optimize their productivity (Balsa-Canto et al., 2005; Banga et al., 2005). In particular, the dynamic optimization of fed-batch reactors has received major attention in recent years (Banga et al., 1997, 2003c) as well as bioprocesses related to the food industry (Banga et al., 2003b; García et al., 2006).

The mathematical formulation of the dynamic optimization problem can be stated as:

$$\min_{\mathbf{u}(t),\mathbf{v},t_f} C\left(\mathbf{x}(t_f),\mathbf{z}(t_f),\mathbf{u}(t_f),\mathbf{v},t_f\right) \tag{1.1}$$

subject to the system dynamics:

$$\dot{\mathbf{x}} = F\left(\mathbf{x}(t),\mathbf{z}(t),\mathbf{u}(t),\mathbf{v},t\right) \tag{1.2}$$

$$\mathbf{x}(0) = \mathbf{x}_0 \tag{1.3}$$

$$\mathbf{u}(0) = \mathbf{u}_0 \tag{1.4}$$

$$\mathbf{z}(0) = \mathbf{z}_0 \tag{1.5}$$

where $\mathbf{x}(t) \in \mathbf{X} \subset \mathbb{R}^n$ and $\mathbf{z}(t) \in \mathbf{Z} \subset \mathbb{R}^m$ are the vectors of differential and algebraic states respectively; $\mathbf{u}(t) \in \mathbf{U} \subset \mathbb{R}^p$ is the vector of control (input) variables; $\mathbf{v} \in \mathbf{V} \subset \mathbb{R}^q$ are time invariant parameters; t is the time (and t_f is the final time); C is a functional to be minimized; F is the set of differential-algebraic equations describing the systems dynamics; finally, $\mathbf{x}_0$, $\mathbf{z}_0$, and $\mathbf{u}_0$ are the values of the respective vectors at the initial time, t_0.

Equality and inequality constraints may be imposed. Some of them must be satisfied over the whole process time (path constraints):

$$\mathbf{H}_{path}\left(\mathbf{x}(t),\mathbf{z}(t),\mathbf{u}(t),\mathbf{v},t\right) = 0 \qquad \forall t \tag{1.6}$$

$$\mathbf{G}_{path}\left(\mathbf{x}(t),\mathbf{z}(t),\mathbf{u}(t),\mathbf{v},t\right) \leq 0 \qquad \forall t \tag{1.7}$$

while others must be only satisfied at the end of the process (endpoint constraints):

$$\mathbf{H}_{end}\left(\mathbf{x}(t_f),\mathbf{z}(t_f),\mathbf{u}(t_f),\mathbf{v},t_f\right) = 0 \tag{1.8}$$

$$\mathbf{G}_{end}\left(\mathbf{x}(t_f),\mathbf{z}(t_f),\mathbf{u}(t_f),\mathbf{v},t_f\right) \leq 0 \tag{1.9}$$

The control variables and/or the time-invariant parameters may be subject to lower and upper bounds:

$$\mathbf{u}^L \leq \mathbf{u}(t) \leq \mathbf{u}^U \tag{1.10}$$

$$\mathbf{v}^L \leq \mathbf{v} \leq \mathbf{v}^U \tag{1.11}$$

1.1.2 Integrated design and control

The general statement of the simultaneous (integrated) design and control problem takes into account the process and control superstructures, indicating the different design alternatives (Schweiger and Floudas, 1997; Bansal et al., 2000; Sakizlis et al., 2004). This general approach results in mixed integer optimal control problems. The aim is to simultaneously find the static variables of the process design as well as the operating conditions and the controllers' parameters which optimize a combined measure of the plant economics and its controllability, subject to a set of constraints which ensure appropriate dynamic behavior and process specifications. In recent years, several authors have stated the necessity of performing simultaneous design and control in bioprocess engineering (Bogle et al., 1996; Groep et al., 2000; Banga et al., 2003c; Moles et al., 2003a). We state our problem as follows:

Find $\mathbf{v}$ to minimize:

$$C = \sum w_i \cdot \phi_i \tag{1.12}$$

subject to

$$\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{v}, t) = 0 \tag{1.13}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{1.14}$$

$$\mathbf{h}(\mathbf{x}, \mathbf{v}) = 0 \tag{1.15}$$

$$\mathbf{g}(\mathbf{x}, \mathbf{v}) \leq 0 \tag{1.16}$$

$$\mathbf{v}^L \leq \mathbf{v} \leq \mathbf{v}^U \tag{1.17}$$

where $\mathbf{v}$ is the vector of decision variables (e.g., design variables, operating conditions, parameters of controllers, set points, etc.); C is the cost (objective function) to minimize (normally a weighted combination of capital, operation and controllability costs, ϕ_i); $\mathbf{f}$ is a functional for the system dynamics (i.e., the nonlinear process model); $\mathbf{x}$ is the vector of the states (and $\dot{\mathbf{x}}$ is its derivative); t_0 the initial time for the integration of the system of

differential algebraic equations (and, consequently, $\mathbf{x}_0$ is the vector of the states at that initial time); $\mathbf{h}$ and $\mathbf{g}$ are possible equality and inequality path and/or point constraint functions which express additional requirements for the process performance; and, finally, $\mathbf{v}^L$ and $\mathbf{v}^U$ are the upper and lower bounds for the decision variables. The dependence of ϕ_i upon the decision variables, $\mathbf{v}$, is defined by the problem formulation. In some cases this dependence is simple and straightforward (i.e., when minimizing the cost of a chemical process, one of the ϕ_i can be equal to a reactor size, which may also be a decision variable), whereas in others there might not be an explicit expression for this dependence (i.e., ϕ_i can be the integral square error, *ISE*, of a controller which, in general, does not depend explicitly on the decision variables).

1.1.3 Parameter estimation

Rigorous dynamic mathematical models are essential for the optimization and on-line control of industrial bioprocesses (Vanrolleghem and Dochain, 1998). Model building consists of different stages: First, based on the theoretical or empirical knowledge about the process, the objectives are set; in a second stage, the system must be identified from the experimental data. The parameters are estimated and, finally, the model can be validated (Rodríguez-Fernández, 2006). The estimation of the model parameters is usually formulated as an optimization problem. In the case of nonlinear dynamic models (and considering continuous measurements over the time), we state the problem as follows:

Find $\mathbf{p}$ to minimize:

$$J = \int_0^{t_f} \left(y_{msd}(t) - y(\mathbf{p},t)\right)^T W(t)\left(y_{msd}(t) - y(\mathbf{p},t)\right)dt \tag{1.18}$$

subject to

$$\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{y}, \mathbf{p}, \mathbf{v}, t) = 0 \tag{1.19}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{1.20}$$

$$\mathbf{h}(\mathbf{x}, \mathbf{y}, \mathbf{p}, \mathbf{v}) = 0 \tag{1.21}$$

$$\mathbf{g}(\mathbf{x}, \mathbf{y}, \mathbf{p}, \mathbf{v}) \leq 0 \tag{1.22}$$

$$\mathbf{p}^L \leq \mathbf{p} \leq \mathbf{p}^U \tag{1.23}$$

where J is the cost function to be minimized; $\mathbf{p}$ is the set of model parameters to be estimated; y_{msd} are values of the experimentally measured system state variables; $y(\mathbf{p},t)$ are the values

of the state variables predicted by the model; $W(t)$ is a scaling matrix; $\mathbf{x}$ are the differential state variables and $\mathbf{v}$ is a vector of parameters; $\mathbf{f}$ describes the system dynamics; $\mathbf{h}$ and $\mathbf{g}$ are possible equality and inequality path and/or point constraint functions which express additional requirements for the system behavior; finally, $\mathbf{p}^L$ and $\mathbf{p}^U$ are the upper and lower bounds for the parameters.

1.2 Global optimization methods in bioprocess optimization

Model based optimization can be successfully used to improve the design and/or operation of single units or full process plants. Typically, most of the problems in bioprocess engineering applications are highly constrained and exhibit nonlinear dynamics. These properties often result in non-convexity and multimodality. Furthermore, in many complex process models some kind of noise and/or discontinuities (either due to numerical methods, or to intrinsic properties of the model) is present. Therefore, there is a great need of robust global optimization solvers which can locate the vicinity of the global solution in a reasonable number of iterations and handle noise and/or discontinuities. In general, (iterative) gradient-based local methods for constrained nonlinear programming (NLP) problems are very efficient, but they can only handle differentiable objective and constraint functions based on a set of continuous variables (Gill et al., 1989). Additionally, convergence to the global solution is not guaranteed in the case of multimodal problems. Therefore, one must use the so-called global optimization (GO) methods (Moles et al., 2003a; Banga et al., 2003b,a) in order to provide an approximation of the global optimum. GO methods can be roughly classified as being deterministic (Grossmann, 1996; Floudas and Pardalos, 2000) or stochastic (Guus et al., 1995).

1.2.1 Deterministic GO methods

These methods, also called exact methods, assure convergence to the global optimum for certain problems, although no algorithm can solve general GO problems with certainty in polynomial time. For these methods, the computational effort usually increases exponentially with the problem size. Further, they have requirements (e.g., smoothness, differentiability) which are rarely met in realistic applications, although very significant advances have been recently made (Esposito and Floudas, 2000b; Singer et al., 2001; Papamichail and Adjiman, 2002). Reviews of these methods can be found in Pinter (1996) and Floudas (2000).

1.2.2 Stochastic GO methods

Stochastic GO methods can usually find solutions close to the global solution in relatively short computation times compared to deterministic GO methods (Banga et al., 1997; Ali et al., 1997; Banga et al., 2003a; Moles et al., 2003a,b). Note that with these stochastic methods, the convergence to global optimality is not guaranteed, but many empirical studies have shown that they are often the best methods for many classes of problems. Another advantage of these methods is that they are easy to implement, and they can treat the objective function as a black box (i.e., a simple connection between inputs and outputs, with no derivative information needed). This feature is specially appealing in the case of complex dynamic systems where the objective function is the result of e.g., a simulation provided by a third-party software with restricted access for the user (a common case in real industrial applications). The number of stochastic algorithms has rapidly increased due to the difficulties of solving complex optimization problems by traditional methods. The most important groups of stochastic algorithms are:

- **Random search and adaptive methods:** They were developed in the 50's and 60's (Brooks, 1958; Matyas, 1965; Rastrigin and Rubinstein, 1969). One of the most popular adaptive methods is the controlled random search (CRS) by Price (1983), which basically consists in generating solutions within the search space and replacing them by other solutions improving their function values by an iterative procedure. The main drawback of these algorithms is their slow convergence rate towards the global optimum for high dimensional problems.

- **Clustering methods:** These methods work clustering solutions based on some characteristics (normally taking into account some kind of distance among them) to create groups. They normally use these clusters to perform *multi-start* procedures avoiding to use similar initial points, thus being more efficient than classical *multi-start* (Törn, 1973; Rinnooy-Kan and Timmer, 1987).

- **Metaheuristics:** A meta-heuristic is an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces using learning strategies to structure information in order to find efficiently near-optimal solutions (Osman and Kelly, 1996). Chapter 2 is dedicated to this type of optimization algorithms.

1.2.3 Hybrid methods

The key concept of hybrid methods is synergy. A hybrid method tries to exploit the best properties of different methodologies (Fraga, 2006). A hybrid method may consist of a global method coupled with a local search (Csendes, 1988; Chelouah and Siarry, 2003; Fraga and Žilinskas, 2003; Rodríguez-Fernández et al., 2006b; Egea et al., 2007a; Lasdon and Plummer, 2008), a stochastic GO method combined with a deterministic one (Balsa-Canto et al., 2005) or a combination of two or more stochastic GO methods (Preux and Talbi, 1999). Hybrid methods can be classified according to the type of hybridization they use: sequential or parallel. In sequential hybridization, two or more algorithms are applied one after another, using as starting conditions the results obtained by the previous algorithms. Some examples of sequential hybridization successfully applied to bioprocess engineering optimization can be found in Banga and Seider (1996); Banga et al. (2005); Rodríguez-Fernández et al. (2006b). For this type of hybridization, a key issue is to the decide the amount of search to be performed by each method. In parallel hybridization we can distinguish between synchronic and asynchronic parallel hybridization. In synchronic parallel hybridization one algorithm is used as an operator of other(s) (Chelouah and Siarry, 2003; Egea et al., 2007a). Asynchronic parallel hybridization implies a cooperative design in which two or more algorithms exchange information to increase their respective performances (Gras et al., 2003; Vrugt and Robinson, 2007).

1.2.4 Surrogate-based global optimization

In general, all the GO approaches presented above require a significant number of evaluations of the objective and constraint functions. In the case of realistic problems, these models may be costly to evaluate, posing a major challenge to the application of GO methods. In recent years, a number of approaches have been proposed to obtain surrogate models which are cheaper to evaluate than the original ones in terms of the computational CPU time, and which imitate the original model based on a reduced number of sampled points or simulations. Provided the surrogate model is accurate enough, the computation times can be significantly reduced. Surrogate model-based methods can be classified into two groups: non interpolating (e.g., quadratic polynomials and other regression models) and interpolating methods (e.g., basis functions and kriging). At the same time, both methods can be one-stage or two-stage methods. Two-stage methods fit first a response surface using sample points

from the real model and then optimize an auxiliary function based on the fitted surface. A potential disadvantage of these methods is that the initial surface may not accurately fit the real model, which can cause the optimization to stop prematurely, or converge to non-global solutions. On the other hand, one-stage methods evaluate hypotheses about the location of the optimum. This is done by examining the best-fitting response surface passing through the observed data and other points in which the optimum is presumed to be located. Each hypothesis is evaluated and the surface is constructed by evaluating the new points where this credibility is maximum.

Jones (2001b) presented a taxonomy of these methods with a complete overview of the different approaches in which it is concluded that interpolating methods are more suitable than non-interpolating methods to locate the global optimum, specifically mentioning kriging and basis functions. However, non-interpolating quadratic fitting methods have been widely used in response surfaces-based engineering design as surveyed in Simpson et al. (2001).

Chapter 2

Metaheuristics for global optimization

The term metaheuristic, originally introduced by Fred Glover at the same time that tabu search (Glover, 1986), is composed by the prefix *meta-* (in Greek, *beyond*) and heuristic (in Greek, *to find*). In computer science, a heuristic is a procedure to provide high quality solutions in short computational time to a hard optimization problem. A heuristic is not usually based on a formal analysis but arises from an expert knowledge of the task to be solved.

There are some definitions of metaheuristics (Osman and Laporte, 1996; Osman and Kelly, 1996; Blum and Roli, 2003; Dréo et al., 2006). The one given by Dorigo and Stützle (2004) is shown below:

> *a metaheuristic can be seen as a general-purpose heuristic method designed to guide an underlying problem-specific heuristic (...) A metaheuristic is therefore a general algorithmic framework which can be applied to different optimization problems with relative few modifications to make them adapted to a specific problem.*

2.1 Desirable properties of a metaheuristic

Melián et al. (2003) proposed a list of properties that a metaheuristic should comply with. It must be noted that some of these properties are opposite to others, therefore it is not possible

to meet all of them at the same time. These properties are the following:

- **Simplicity:** The metaheuristic must be based on a simple and clear principle.
- **Accuracy:** It must be formulated with accurate terms.
- **Consistency:** The elements of the metaheuristic must be deduced from its principles.
- **Effectiveness:** It must provide high quality solutions (the global optimum or solutions close to it) with high probability.
- **Efficiency:** It must not employ a huge amount of computational resources (e.g., computation time and memory).
- **Generality:** It must be applicable to a large variety of problems.
- **Versatility:** It must be adaptable to different contexts of model changes.
- **Robustness:** It must not be too sensitive to small modifications in the model or in the application context.
- **Interactivity:** It must allow the user to apply his/her own knowledge of the problem to improve the performance.
- **Multiplicity:** It must provide different high quality solutions among which the user can choose.
- **Autonomy:** The metaheuristic must work without adjusting any parameter if necessary.

2.2 Types of metaheuristics

There are several classifications of metaheuristics according to different criteria (see, for example, Taillard et al. 2001; Dréo et al. 2007). We have found the most complete classification in the web page maintained by Johann Dréo[1]. Here we will distinguish two main criteria to classify metaheuristics (see Figure 2.1):

[1] http://nojhan.free.fr/metah/

- Metaheuristics based on a population or set of solutions (e.g., genetic algorithms, evolution strategies, differential evolution, scatter search, ant colony optimization, particle swarm optimization, memetic algorithms, estimation of distribution algorithms) or based on a trajectory (e.g., tabu search, simulated annealing, GRASP, iterated local search, hill climbing).

- Metaheuristics inspired in nature (e.g., genetic algorithms, evolution strategies, ant colony optimization, particle swarm optimization, simulated annealing) or not (e.g., iterated local search, GRASP, scatter search, tabu search, differential evolution, memetic algorithms, hill climbing, estimation of distribution algorithms).

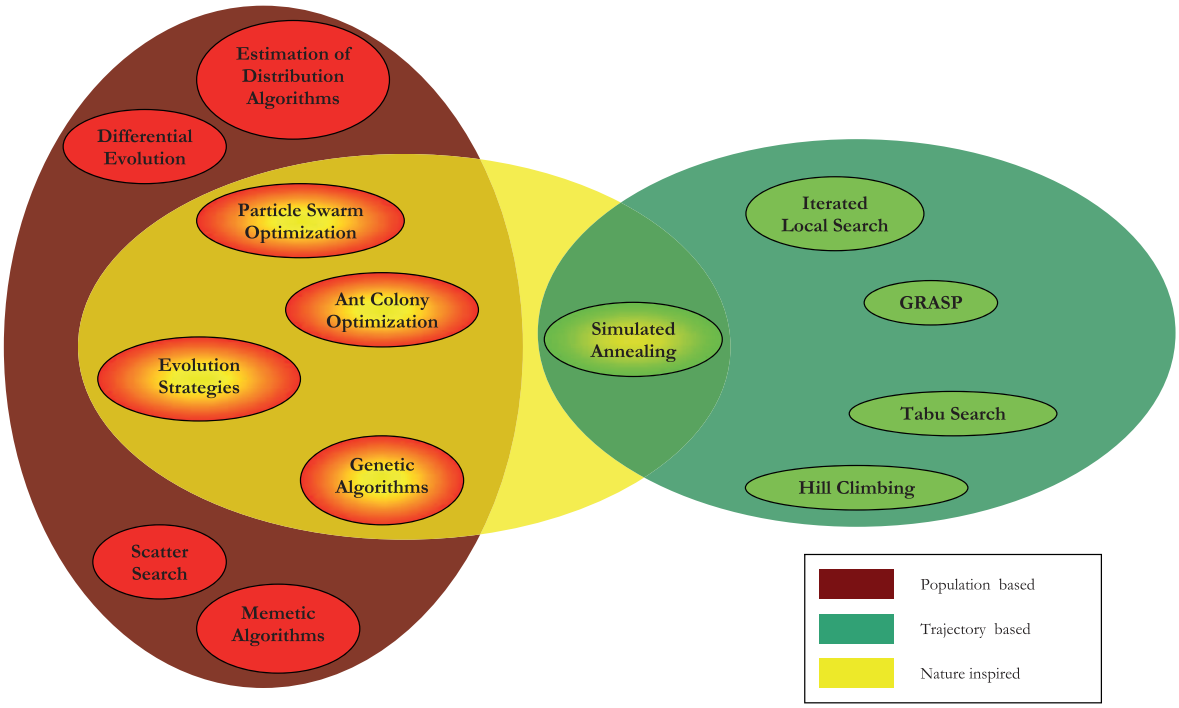


Figure 2.1: Taxonomy of metaheuristics

Most metaheuristics were initially developed for combinatorial problems. Since optimization problems arising in bioprocess engineering are usually continuous or mixed-integer, we must use specific adaptations to this context. There are some studies and adaptations of metaheuristics to continuous problems (Hedar, 2004; Michalewicz and Siarry, 2008) even if many of them have not extensively been applied to bioprocess engineering optimization.

In the following lines we will briefly introduce some of the most well-known metaheuristics. Some examples of their application to process engineering optimization will be shown. Scatter search will be more deeply analyzed in Chapter 3.

2.2.1 Genetic Algorithms (GAs)

Genetic algorithms were developed in the 70's by Holland (1975) and improved by Goldberg (1989). They are inspired on principles of natural selection and genetics. GAs encode the decision variables in sets of solutions called *chromosomes*, formed by different parts (*genes*) with some values (*alleles*). GAs are population-based algorithms, in which the population size is usually an important factor affecting their performance and scalability. Once a problem is encoded in *chromosomes* and a fitness measure for selecting good solutions (usually the objective function value) has been chosen, the population can start to evolve using steps shown in Algorithm 1 (Sastry and Goldberg, 2005):

Algorithm 1 Genetic Algorithm pseudocode

Initialization: Generate the initial population over the search space
Evaluation: Evaluate the fitness of the initial population
repeat
  Selection: Assign a higher probability of being subject to the next steps to the best solutions
  Recombination: Combine parts of two or more parental solutions to create new ones
  Mutation: Modify locally a solution (usually in a random way)
  Replacement: The offspring created by selection, recombination and mutation replaces the original parental population
until Termination criterion is met

Due to their popularity, GAs have been widely applied to many optimization problems, including process engineering. Some recent examples of their application can be found in Fraga and Senos Matias (1996); Garrard and Fraga (1998); Sarkar and Modak (2004); Wang et al. (2004) and Ponsich et al. (2007).

2.2.2 Evolution Strategies (ES)

Evolution strategies were proposed in the 60's and 70's by Ingo Rechenberg (Rechenberg, 1973). ES use natural problem-dependent representations, and primarily mutation and selection as search operators. The operators are applied in a loop. An iteration of the loop is called a generation. The sequence of generations is continued until a termination criterion is met. The first formulated strategy only considered one parental solution and one offspring solution ((1+1)-strategy). The first population-based strategy was the so-called (μ+1)-strategy in which there is a population of μ parents that are combined to generate a child (which could also be mutated). The child replaces the worst parent as long as the former outperforms the latter. Later, other strategies which constitute the current state-of-the-art were formulated ($(\mu + \lambda)$ and (μ, λ) strategies). These new formulations allowed the

algorithms to be parallelized and parameter self-adjustable. In both strategies, μ parents generate λ children by recombination and mutation. In the (μ, λ)-strategy only the best μ children pass to the next generation whereas none of the parents are kept. This is a non elitist strategy, since it allows the population to decrease its average quality in each generation. In the $(\mu + \lambda)$-strategy the best μ members of the union between parents and children are selected for the next generation. This can accelerate the convergence rate but also make the algorithm converge prematurely to sub-optimal solutions. Algorithm 2 shows the pseudocode for (μ, λ) and $(\mu + \lambda)$ strategies.

Algorithm 2 Evolution Strategy pseudocode

```
Generate a set of solutions over the search space
Select the best μ elements among the set of solutions = Pop
repeat
    Mutate the elements in Pop to create an offspring of λ elements
    (μ, λ): Select the best μ elements of the offspring to create the new Pop
    (μ + λ): Select the best μ elements from the union of Pop and the offspring to create the new Pop
until Termination criterion is met
```

Unlike other metaheuristics, ES have a solid theoretical basis (Beyer, 1996; Beyer and Schwefel, 2002). Together with GAs, ES have been one of the most applied metaheuristics to process engineering optimization (Roubos et al., 1999; Costa and Oliveira, 2001; Banga et al., 2003b,c; Moles et al., 2003a,b; Park and Lee, 2004; Banga et al., 2005; Balku and Berber, 2006).

2.2.3 Differential Evolution (DE)

This algorithm (Storn and Price, 1997) is an evolutionary algorithm which uses vector differences for perturbing the vector population. The algorithm makes use of two operators: The *differential mutation* (i.e., perturb a vector by adding to it the difference of other two population vectors multiplied by a factor) and the *crossover* (i.e., exchange the value of some decision variables between the original and the perturbed vector with a certain probability). The pseudocode of the algorithm is shown in Algorithm 3.

DE has been widely used by the scientific community in many research areas (Price et al., 2005), being currently one of the most popular algorithms for global optimization. Some recent applications of DE to process engineering problems can be found in Wang et al. (2001); Banga et al. (2003c); Babu and Angira (2006); Angira and Santosh (2007).

Algorithm 3 Differential Evolution pseudocode

```
Set algorithm parameters, F and CR
Initialize and evaluate population P
while Termination criterion not met do
  for i = 1 to NP do
    Choose randomly x_j and x_k with i ≠ j ≠ k
    MUTATION: u_i = x_i + F(x_j − x_k)
    for n = 1 to prob_size do
      Generate a random number within [0, 1], nrand
      if nrand ≥ CR then
        CROSSOVER: v_{i,n} = u_{i,n}
      else
        v_{i,n} = x_{i,n}
      end if
    end for
    if v_i is better than x_i then
      Replace x_i by v_i
    end if
  end for
end while
```

2.2.4 Tabu Search (TS)

TS was created in the 70's by Fred Glover (Glover, 1977) although the formal name and the methodology were established later (Glover, 1989, 1990). TS is based on the premise that problem solving, in order to qualify as intelligent, must incorporate *adaptive memory* and *responsive exploration*. The adaptive memory feature of TS allows the implementation of procedures that are capable of searching the solution space economically and effectively. TS starts from a solution in the search space and makes a movement to another solution within its neighborhood. TS begins in the same way as ordinary local or neighborhood search, proceeding iteratively from one point (solution) to another until a chosen termination criterion is satisfied. We may contrast TS with a simple descent method where the goal is to minimize the objective function, $f(x)$. Such a method only permits moves to neighbor solutions that improve the current objective function value and ends when no improving solutions can be found. The final x obtained by a descent method is called a local optimum, since it is better than all solutions in its neighborhood, or, at least, as good as them. The evident shortcoming of a descent method is that such a local optimum in most cases will not be a global optimum.

TS permits moves that deteriorate the current objective function value but the moves are chosen from a modified neighborhood, $N^*(x)$. Short and long-term memory structures are responsible for the specific composition of $N^*(x)$. In other words, the modified neighborhood

is the result of maintaining a selective history of the states encountered during the search. In the TS strategies based on short-term considerations, $N^*(x)$ characteristically is a subset of $N(x)$, and the tabu classification serves to identify elements of $N(x)$ excluded from $N^*(x)$. In TS strategies that include longer term considerations, $N^*(x)$ may also be expanded to include solutions not ordinarily found in $N(x)$, such as solutions found and evaluated in past search, or identified as high quality neighbors of these past solutions. Characterized in this way, TS may be viewed as a dynamic neighborhood method (Glover et al., 2007). This means that the neighborhood of x is not a static set, but rather a set that can change according to the history of the search.

A tabu solution (or area) remains in a *tabu list* during a defined number of iterations (*tabu tenure*). TS uses the concept of *aspiration criterion* which in a simple form can be defined as follows: "if a solution outperforms the best solution found so far, the search will be directed to it regardless its tabu status". Apart from the short-term memory, tabu search also makes use of a long-term memory which keeps track of the frequencies or attributes of the visited solutions to identify different regions. The long-term memory has two associated strategies: intensification and diversification. Intensification consists in revisiting explored areas to investigate them more deeply. Areas containing good solutions are favored. Diversification consists in visiting new areas not yet explored. Algorithm 4 shows the pseudocode of a basic TS procedure.

Algorithm 4 Tabu Search pseudocode

```
Set x_0
x_c = x^best = x_0
Tabu list=∅
repeat
  Mutate x_c to create x_new
  if x_new is in Tabu list then
    if aspiration criterion met then
      x_c = x_new
      Remove x_new area from the Tabu list
    end if
  else
    x_c = x_new
  end if
  Update Tabu list
  Update x^best
until Termination criterion is met
```

For further information about TS see Glover and Laguna (1997). Some applications to process engineering optimization can be found in Wang et al. (1999); Teh and Rangaiah

(2003); Rajesh et al. (2003); Lin and Miller (2004); Cavin et al. (2005) and Exler et al. (2008).

2.2.5 Particle Swarm Optimization (PSO)

This method, first reported by Kennedy and Eberhart (1995), is based on the idea of social learning and exchange of information among the members of a population. Each member of the swarm (i.e., each particle or solution) have two main characteristics: its position and its velocity. A particle changes both parameters by following two solutions: the best solution found so far by the swarm, *Gbest*, and the best solution visited by itself during the search process, *Pbest*.

A pseudocode of the algorithm is shown in Algorithm 5. For further information about swarm optimization see Eberhart et al. (2001). An application of PSO in process engineering can be found in Ourique et al. (2002).

Algorithm 5 Particle Swarm Optimization pseudocode

```
Initialize and evaluate population P
Set velocity of every particle, V, to 0
Set memory of every particle, Pbest = P
Gbest = Best Particle
while Termination criterion not met do
  for i = 1 to NP do
    Set w: inertia weight
    Set C1 and C2: positive constants
    Set Rn1 and R2: random numbers within the interval [0, 1]
    Update speed of each particle according to
    V(i) = w · V(i) + C1 Rn1 (Pbest(i) − P(i)) + C2 Rn2 (Gbest − P(i))
    P(i) = P(i) + V(i)
    Update Pbest and Gbest
  end for
end while
```

2.2.6 Ant Colony Optimization (ACO)

This method, introduced by Dorigo (1992), is based on the social behavior of some insects that present a sophisticated social structure. It was initially developed for combinatorial optimization problems although it has received major attention in recent years for continuous problems (Dréo and Siarry, 2006; Socha and Dorigo, 2008). Its biological basis is the communication established by ants when they seek food. During food seeking, ants modify the environmental conditions by secreting pheromones on their way. Later, other ants will detect the pheromones concentration and will follow those paths more frequently used by

other members of the community, which correspond to the shortest paths from their anthill to the food source. The ants movements are always randomized, but the probability of flitting into the direction of pheromones is higher. Some important parameters to be taken into account are: (i) the pheromone evaporation, to avoid following the initial paths which were completely random, and (ii) the size of the colony: a small size may not provide enough information about good paths whereas a big size may increase the computational effort too much. It is worth noting that the use of pheromones can be interpreted as an implementation of a memory structure. This fact together with the introduction of probabilistic elements allow us to consider ACO as a particular and efficient case of probabilistic TS. Algorithm 6 shows the pseudocode of a basic ACO procedure.

Algorithm 6 Ant Colony Optimization pseudocode

```
Generate a set of solutions over the search space
Select the best k elements among the set of solutions as the set of ants, s
repeat
   Build pheromones from ants in s
   Create new solutions according to pheromones information
   Take the best k elements among s and the new solutions as new s
until Termination criterion is met
```

For further information about ACO see Dorigo and Stützle (2004). Some examples of application of ACO in process engineering can be found in Jayaraman et al. (2000); Rajesh et al. (2001); Chunfeng and Xin (2002); Zhang et al. (2005).

2.2.7 Simulated Annealing (SA)

Introduced by Kirkpatrick et al. (1983), this is one of the most popular methods among the global optimization community. In SA, the process of slow and controlled cooling of a melted material to obtain a crystalline structure is reproduced. This structure corresponds to a minimum of energy (represented by the function value in optimization). Starting from a given temperature, a new solid state is generated by applying a perturbation. The energy for this new state is evaluated. If the new solid state has a lower energy than the previous one, the movement is automatically accepted; otherwise it is accepted with a probability P given by

$$P = e^{-\frac{\Delta E}{K_B T}} \tag{2.1}$$

where E is the energy, K_B is the Boltzmann constant and T is the temperature. The

probability P decreases along the optimization procedure. The critical point of the algorithm is the definition of the cooling scheme. Algorithm 7 shows the SA pseudocode.

Algorithm 7 Simulated Annealing pseudocode

```
Set x_0,   iter = 0
x_c = x^best = x_0
Create x_new randomly
repeat
  ΔE = f(x_new) − f(x_c)
  if ΔE < 0 then
    x_c = x_new
    if f(x_c) < f(x^best) then
      x^best = x_c
    end if
  else
    T = φ(iter)
    rnd =random number
    if rnd < e^(−ΔE/(K_B T)) then
      x_c = x_new
    end if
  end if
  Mutate x_c to create a new x_new
  iter = iter + 1
until Termination criterion is met
```

SA is quite a popular algorithm within the optimization community. Thus, there are several studies in different fields in which we can find guidelines to set the search parameters, including process engineering (Kookos, 2004; Faber et al., 2005; Sun and Lin, 2006).

2.2.8 Memetic Algorithms (MAs)

The basic scheme of MAs is a combination of a local search with a crossover operator, an exact method or other heuristics. They present faster convergence rates than other evolutionary algorithms for some problems in which a local search procedure is efficient. The term *Memetic Algorithms* appeared in 1989 (Moscato, 1989). The pseudocode of a basic MA is given in Algorithm 8.

Algorithm 8 Memetic Algorithm pseudocode

```
Generate an initial population
repeat
  Recombine, mutate or apply other operator(s) among the population members
  Improve population members (or a subset of it) with local search
  Select new population for the next generation
until Termination criterion is met
```

Although this recent method has been mainly applied in combinatorial optimization prob-

lems, some engineering applications have arisen recently (Zelinka et al., 2001; Benali et al., 2007). This type of algorithms have been also proposed for those optimization problems involving computationally expensive simulation models (typical of complex bioprocess) to intensify the search in promising areas when the budget of function evaluations is small (Zhou et al., 2007). However, this strategy could lead to suboptimal solutions if the number of simulations allowed is not large enough.

2.2.9 Iterated Local Search (ILS)

Iterative local search consists of two phases: the first one in which a solution is generated and a second one in which that solution is improved. Every iteration produces a solution (usually a local optimum) and the best of them is the algorithm output.

Termination of local search can be based on a time bound. Another common choice is to terminate when the best solution found by the algorithm has not been improved in a given number of steps. Local search algorithms are typically incomplete algorithms, as the search may stop even if the best solution found by the algorithm is not optimal. This can happen even if termination is due to the impossibility of improving the solution, as the optimal solution can lie far from the neighborhood of the solutions crossed by the algorithms. The pseudocode of a basic ILS is shown in Algorithm 9.

Algorithm 9 Iterated Local Search pseudocode

$x^{best} = \emptyset$
repeat
  Create a random solution, x_r, within the search space
  Perform a local search using x_r as initial point
  Save the local solution found
  Update x^{best}
until Termination criterion is met

Many authors have added several features to this basic scheme such as memory, clustering methods and others to make the search more efficient. In many cases, the addition of these features creates a new metaheuristic, as is the case of GRASP, presented in Section 2.2.10. For further information about local search methods see Martí (2003). An example of application of this metaheuristic in bioprocess engineering optimization can be found in Rodríguez-Acosta et al. (1999).

2.2.10 GRASP

GRASP (*Greedy Randomized Adaptive Search Procedures*) is a multistart local search procedure, where each iteration consists of two phases: a construction phase and a local search phase. The GRASP methodology was developed in the late 1980s, and the acronym was coined by Feo and Resende (1995). It was first used to solve computationally-difficult set covering problems (Feo and Resende, 1989). Each GRASP iteration consists in constructing a trial solution and then applying an exchange procedure to find a local optimum (i.e., the final solution for that iteration). The construction phase is iterative, greedy, and adaptive. It is iterative because the initial solution is built considering one element at a time. It is greedy because the addition of each element is guided by a greedy function. It is adaptive because the element chosen at any iteration in a construction is a function of those previously chosen (that is, the method is adaptive in the sense of updating relevant information from one construction step to the next). The improvement phase typically consists of a local search procedure.

Performing multiple GRASP iterations may be interpreted as a means of strategically sampling the solution space. Based on empirical observations, it has been found that the sampling distribution generally has a mean value that is inferior to the one obtained by a deterministic construction, but the best over all trials dominates the deterministic solution with a high probability. The intuitive justification of this phenomenon is based on the ordering statistics of sampling. GRASP implementations are generally robust in the sense that it is difficult to find or devise pathological instances for which the method will perform arbitrarily bad. The robustness of this method has been well documented in applications to production, flight scheduling, equipment and tool selection, location, and maximum independent sets. The pseudocode of a basic GRASP implementation is shown in Algorithm 10.

Algorithm 10 GRASP pseudocode

$x^{best} = \emptyset$
init_points = *local_solutions* = $\emptyset$
repeat
  Construction phase: Select a suitable initial point, x_0 based on the information provided by *init_points* and *local_solutions*
  Perform a local search using x_0 as initial point
  Save both x_0 and the local solution in *init_points local_solutions* respectively
  Update x^{best}
until Termination criterion is met

GRASP was originally designed for combinatorial optimization problems, although an adaptation to continuous problems has recently appeared (Hirsch et al., 2007).

2.2.11 Hill Climbing

Hill climbing is an optimization technique which belongs to the family of local search. In hill climbing, the basic idea is to always head towards a state which is better than the current one (Rich and Knight, 1991). The algorithm is started from a random solution. It sequentially makes small changes to the solution, each time improving it a little bit. At some point the algorithm can not see any improvement anymore; then, the algorithm terminates. Ideally, at that point a solution is found that is close to optimal, but it is not guaranteed that hill climbing will ever come close to the optimal solution. Basic hill climbing works as shown in Algorithm 11.

Algorithm 11 Hill Climbing pseudocode

```
Start with current_solution = initial_solution
while not Termination criterion do
  Select a random relative direction from current_solution
  Generate one or more solutions close to current_solution following that direction
  if any of the generated solutions improves current_solution then
    Set this new solution as current_solution
    Continue searching in the same direction
  end if
end while
```

Strategies to avoid cycling or to accelerate the convergence rate can be added in advanced implementations, as in the one by de la Maza and Yuret (1994).

2.2.12 Estimation of Distribution Algorithms (EDAs)

These new algorithms are a variant of evolutionary algorithms which generate new solutions by learning and sampling from the probability distribution of the best individuals of the population at each iteration instead of using crossover and mutation operators (Mühlenbein and Paass, 1996; Larrañaga and Lozano, 2001). With these methods, relationships between decision variables are identified and exploited. Algorithm 12 shows the pseudocode of a basic EDA. Note that it is the same one used for genetic algorithms but changing the steps of recombination and mutation.

A recent application of EDAs to process engineering optimization can be found in Jiang et al. (2006).

Algorithm 12 Estimation of Distribution Algorithm pseudocode

Initialization: Generate the initial population over the search space
Evaluation: Evaluate the fitness of the initial population
repeat
  Calculate a probabilistic model of the population (or a part of it)
  Generate a new population following the probabilistic model
until Termination criterion is met

2.2.13 Hybrid metaheuristics

In recent years, many studies in the field of *hybrid metaheuristics* have been published. A skilled combination of concepts of different metaheuristics can provide a more efficient behavior and a higher flexibility when dealing with real-world and large-scale problems (Talbi, 2002). Some recent examples of hybrid metaheuristics applied to process engineering optimization problems are Chiou and Wang (1999); Srinivas and Rangaiah (2007) and Shelokar et al. (2008).

Chapter 3

Scatter Search

Scatter search is a population-based metaheuristic that has recently been shown to yield promising outcomes for solving combinatorial and nonlinear optimization problems. Based on formulations originally proposed in the 1960s for combining decision rules and problem constraints such as the surrogate constraint method, scatter search uses strategies for combining solution vectors that have proved effective in a variety of problem settings.

3.1 Scatter Search methodology

Scatter search was first introduced by Fred Glover (Glover, 1977) as a heuristic for integer programming. Scatter search orients its explorations systematically, relative to a set of reference points that typically consist of good solutions obtained by prior problem solving efforts. The scatter search template (Glover, 1998) has served as the main reference for most of the scatter search implementations to date. Scatter search methodology is very flexible, since each of its elements can be implemented in a variety of ways and degrees of sophistication. Here we give a basic design to implement scatter search based on the well-known "five-method template" (Laguna and Martí, 2003). The advanced features of scatter search are related to the way these five methods are implemented. That is, the sophistication comes from the implementation of the scatter search methods instead of the decision to include or exclude certain elements (as in the case of tabu search or other metaheuristics).

The fact that the mechanisms within scatter search are not restricted to a single uniform design allows the exploration of strategic possibilities that may prove effective in a partic-

ular implementation. These observations and principles lead to the following "five-method template" for implementing scatter search:

1. A *Diversification Generation Method* to generate a collection of diverse trial solutions.

2. An *Improvement Method* to transform a trial solution into one or more enhanced trial solutions. Neither the input nor the output solutions are required to be feasible, though the output solutions will more usually be expected to be so. If no improvement of the input trial solution results, the "enhanced" solution is considered to be the same as the input solution.

3. A *Reference Set Update Method* to build and maintain a reference set consisting of the b "best" solutions found, where the value of b is typically small compared to the population size of other evolutionary algorithms, organized to provide efficient accessing by other parts of the method. Solutions gain membership to the reference set according to their quality or their diversity.

4. A *Subset Generation Method* to operate on the reference set, to produce several subsets of its solutions as a basis for creating combined solutions.

5. A *Solution Combination Method* to transform a given subset of solutions produced by the *Subset Generation Method* into one or more combined solution vectors.

Figure 3.1 shows the interaction among these five methods and highlights the central role of the reference set (*RefSet*). This basic design starts with the creation of an initial set of solutions P, and then extracts from it the *RefSet*. The darker circles represent improved solutions resulting from the application of the *Improvement Method*.

The *Diversification Generation Method* is used to build a large set P of diverse solutions. The size of P ($PSize$) is typically at least ten times the problem size. The initial *RefSet* is built according to the *Reference Set Update Method*, which can take the b best solutions (as regards their quality in the problem solving) from P to compose the *RefSet*. However, diversity can be considered instead of, or in addition to, quality for the updating. For example, the *Reference Set Update Method* could consist of selecting b distinct and maximally diverse solutions from P. Regardless of the rules used to select the reference solutions, the solutions in *RefSet* are ordered according to quality, where the best solution is the first one in the list. The search is then initiated by applying the *Subset Generation Method* which, in its

simplest form, involves generating all pairs of reference solutions. The pairs of solutions in *RefSet* are selected one at a time and the *Solution Combination Method* is applied to generate one or more trial solutions. These trial solutions are subjected to the *Improvement Method*. The *Reference Set Update Method* is applied once again to build the new *RefSet* with the best solutions, according to the objective function value, from the current *RefSet* and the set of trial solutions. The basic procedure terminates after all the generated subsets are subjected to the *Solution Combination Method* and none of the improved trial solutions are admitted into the *RefSet* under the rules of the *Reference Set Update Method*. However, in advanced scatter search designs, the *RefSet* rebuilding is applied at this point and the best $b/2$ solutions are kept in the *RefSet* while the other $b/2$ are selected from P, replacing the worst $b/2$ solutions, as shown in Figure 3.1. For other possible advanced designs see Martí et al. (2006).

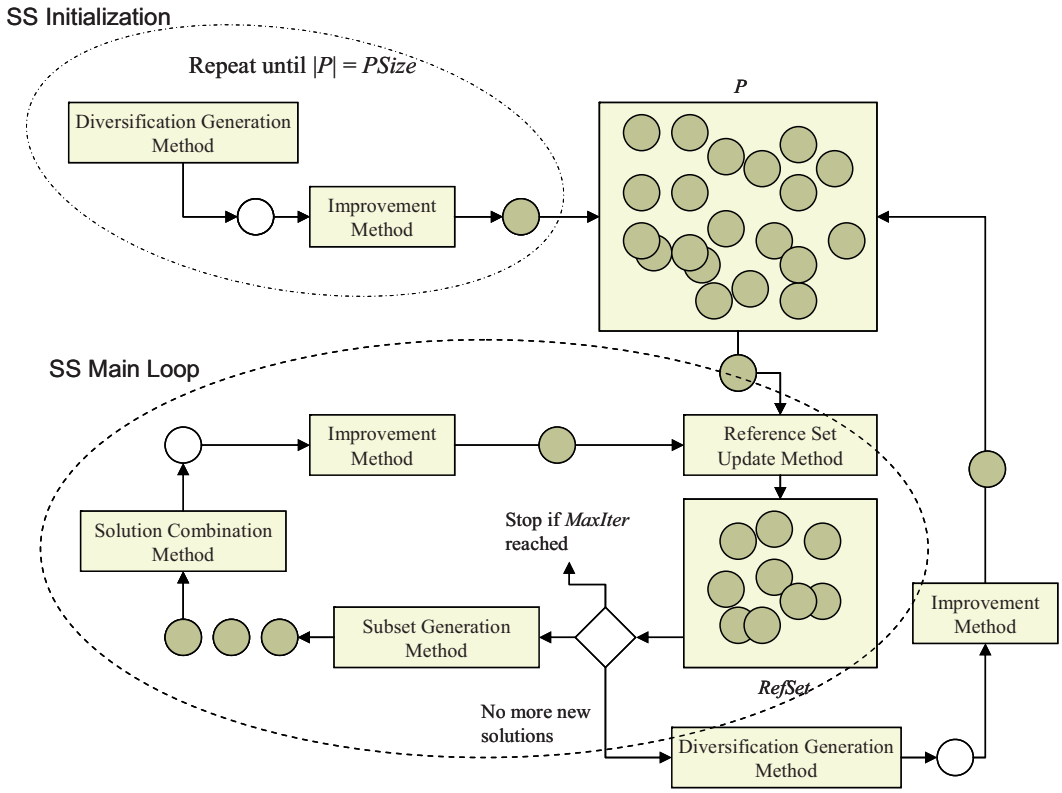


Figure 3.1: Schematic representation of the scatter search design

The *RefSet* is a collection of both high quality solutions and diverse solutions that are used to generate new solutions by way of applying the *Solution Combination Method*. We can use a simple mechanism to construct an initial reference set and then update it during the search. The size of the reference set is denoted by $b = b_1 + b_2$. The construction of the

initial *RefSet* starts with the selection of the best b_1 solutions from P. These solutions are added to *RefSet* and deleted from P. For each solution in the updated P, the minimum of the distances to the solutions in *RefSet* is computed. Then, the solution with the maximum of these minimum distances is selected. This solution is added to *RefSet* and deleted from P, and the minimum distances are updated. The process is repeated b_2 times, where $b_2 = b - b_1$. The resulting *RefSet* has b_1 high quality solutions and b_2 diverse solutions. Algorithm 13 shows a basic scatter search procedure in pseudocode.

Algorithm 13 Basic Scatter Search procedure

```
1: Start with P =Ø
2: repeat
3:    Use the Diversification Generation Method to construct a solution and apply the Improvement
      Method
4:    Let x be the resulting solution
5:    if x ∉ P then
6:       P = P ∪ x
7:    else
8:       Discard x
9:    end if
10: until |P| = PSize
11: Use the Reference Set Update method to build RefSet = {x^1, ..., x^b} with the best b_1 quality
    solutions and b_2 diverse solutions (b_1 + b_2 = b) in P
12: Sort the solutions in RefSet according to their objective function value such that x^1 is the best
    solution and x^b the worst
13: Make NewSolutions = TRUE
14: while NewSolutions do
15:    Generate NewSubsets with the Subset Generation Method
16:    Make NewSolutions = FALSE
17:    while NewSubsets ≠ Ø do
18:       Select the next subset s in NewSubsets
19:       Apply the Solution Combination Method to s to obtain new trial solutions
20:       Apply the Improvement Method to the trial solutions
21:       Apply the RefSet Update Method
22:       if Refset has changed then
23:          make NewSolutions = TRUE
24:       end if
25:       Delete s from NewSubsets
26:    end while
27: end while
```

Of the five methods in scatter search methodology, only four are strictly required. The *Improvement Method* is usually needed if high quality outcomes are desired, but a scatter search procedure can be implemented without it as it occurs in some problems where the *Improvement Method* can not provide high quality solutions due to the problem's nature or when the computation budget is limited to a small number of function evaluations. On the

other hand, hybrid scatter search designs could incorporate a short-term tabu search or other complex metaheuristic (usually demanding more running time).

It is interesting to observe similarities and contrasts between scatter search and the original GA proposals. Both are instances of what are sometimes called "population based" or "evolutionary" approaches. Both incorporate the idea that a key aspect of producing new elements is to generate some form of combination of existing elements. However, GA approaches are predicated on the idea of choosing parents randomly to produce offspring, and further on introducing randomization to determine which components of the parents should be combined. By contrast, the scatter search approach does not emphasize randomization, particularly in the sense of being indifferent to choices among alternatives. Instead, the approach is designed to incorporate strategic responses, both deterministic and probabilistic, that take account of evaluations and history. Scatter search focuses on generating relevant outcomes without losing the ability to produce diverse solutions, due to the way the generation process is implemented.

As other metaheuristics, scatter search has been mainly applied to optimization problems involving integer variables (see Glover et al. 2000a for a list of scatter search applications). However, some adaptations to continuous problems have arisen in recent last years. Fleurent et al. (1996) and Trafalis and Kasap (1996) presented a scatter search approach for continuous optimization. Later, Trafalis and Kasap (2002) combined scatter search with other metaheuristics. Laguna and Martí (2002) presented the OptQuest callable library, based on scatter search, which was used in Ugray et al. (2005). Laguna and Martí (2005) tested some advanced scatter search designs for global optimization and later Herrera et al. (2006) analyzed the performance of different combination methods and improvement strategies. Other advanced implementations have been applied to parameter estimation in systems biology (Rodríguez-Fernández et al., 2006a), chemical and bioprocess optimization (Egea et al., 2007a) and food engineering optimization (Rodríguez-Fernández et al., 2007). Egea et al. (2007b) developed a scatter search-based algorithm for computationally expensive process models.

3.2 Scatter Search tutorial

When presenting an optimization algorithm one usually tries to illustrate the way it works in as much detail as possible. Detailed explanations, pseudocodes, figures and application examples are usually used for this purpose. In Glover et al. (2003) a useful step-by-step

scatter search tutorial for continuous problems is presented. It helps non-experts to start their first implementation. In the next lines, another tutorial of the scatter search basic scheme is presented, based on figures showing solutions over the search space. To follow it, some points must be considered:

1. We will consider a minimization problem.
2. Numbers inside every solution (circles) represent their objective function values.
3. Number of diverse solutions to initiate the procedure, $PSize = 10$.
4. *RefSet* dimension, $b = 4$ solutions.
5. The function to be optimized has 3 minima, 2 of them local (green squares) and 1 global (blue square).
6. For the sake of clarity, the *Improvement Method* is only applied in the last figure of this section.

3.2.1 Initialization

The algorithm starts by creating a set P of *Psize* diverse solutions (10 in our case) with a sampling technique which can be simple randomization, latin hypercube sampling or any other strategy like those proposed in Laguna and Martí (2003). Figure 3.2 shows the initial set of diverse solutions at the beginning of the procedure.

3.2.2 First *RefSet* formation

The initial *RefSet* must contain both quality and diverse solution. The *RefSet Update Method* is called for the first time. Thus, as explained in Section 3.1, $b_1 = 2$ solutions will be selected by quality (i.e., those with the smallest function values since we are dealing with minimization). Then, $b_2 = 2$ solutions must be selected following a diversity criterion. In Figure 3.3, the first b_1 solutions (in red) are automatically selected according to their function value. The next solution (in grey) is selected by maximizing the distance to all the b_1 solutions and the next one (in black) uses the same principle taking into account the b_1 and the rest of solutions already included in b_2. The rest of solutions not included in the *RefSet* are discarded.

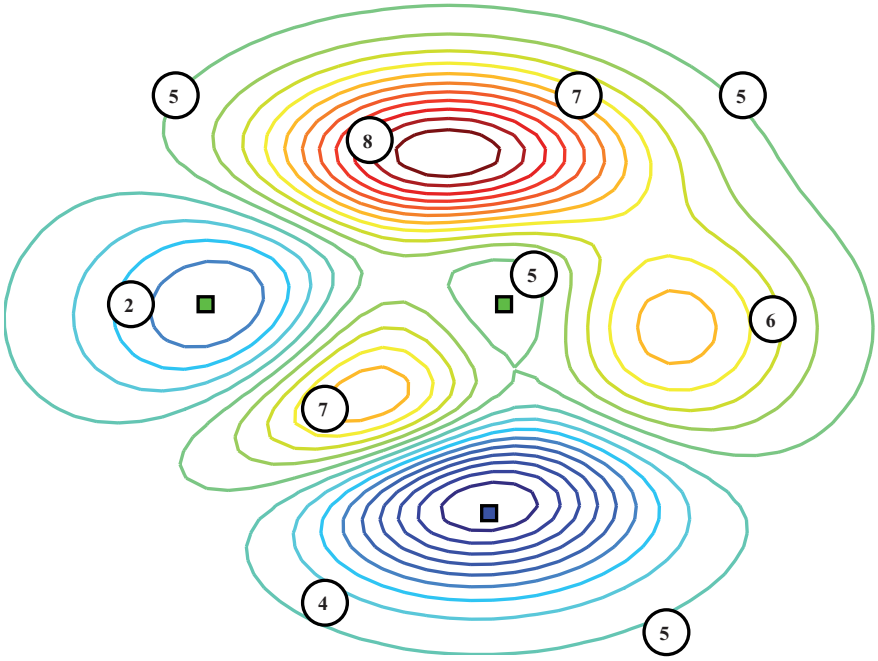


Figure 3.2: Initial set of diverse solutions

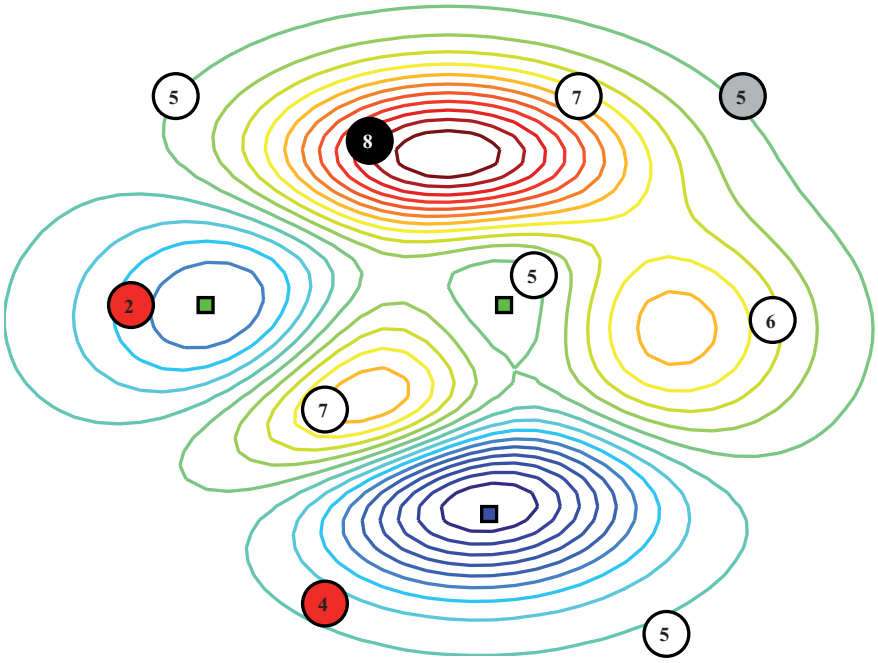


Figure 3.3: First *Refset* formation

3.2.3 Subset generation and combination

Once the initial *RefSet* has been formed, the *Subset Generation Method* creates different sets of solutions to be combined. Then, every set of solutions generates one or more solutions by applying the *Solution Combination Method*. In Figure 3.4, the sets are all the pairs of

solutions in *RefSet*. The *Solution Combination Method*, in this case, consists in generating a solution for each pair inside the segment linking the two solutions of the pair. The generated solutions are represented in blue in Figure 3.4.

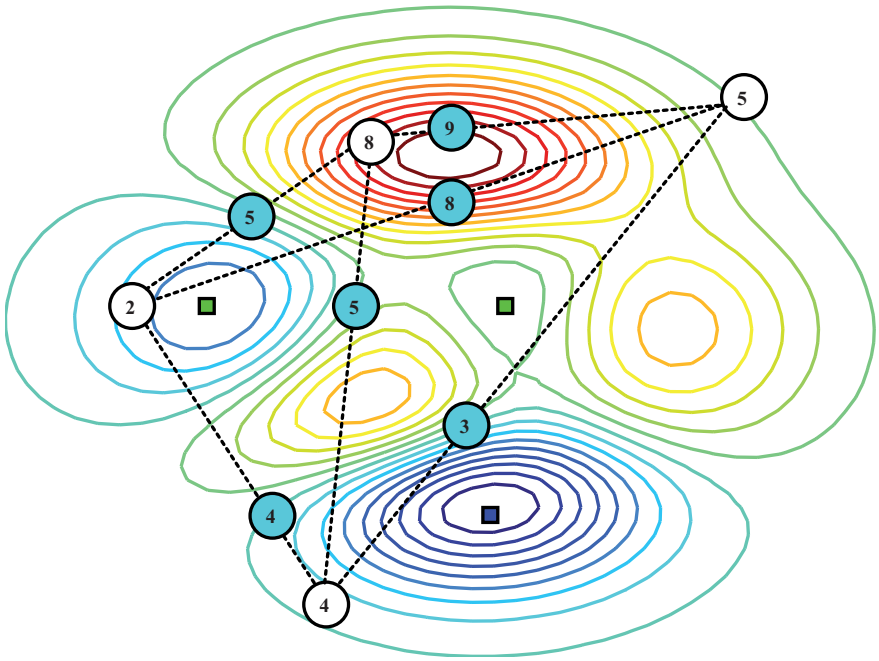


Figure 3.4: Combination of every pair of solutions in *RefSet*

3.2.4 *RefSet* update

After generating new solutions, the *RefSet Update Method* is called again to update the members of the *RefSet*. The update can be done by quality, diversity or a combination of both strategies (Laguna and Martí, 2005). Here we will update the *RefSet* by quality, thus the best b best solutions among the last *RefSet* members and the new generated solutions are selected as the new *RefSet* members. Those solutions with the minimum function values are selected. In our example, the new *RefSet* is composed by 2 new solutions and 2 solutions which were part of the *RefSet* in the previous iteration (see Figure 3.5).

3.2.5 *RefSet* regeneration

The previous steps, including the *Improvement Method*, which will be illustrated in Section 3.2.6, are repeated until a termination criterion is met. Scatter search usually makes use of a memory element to avoid performing combinations among sets of solutions already combined. It may occur that, at some point, no new subsets are available. At this moment,

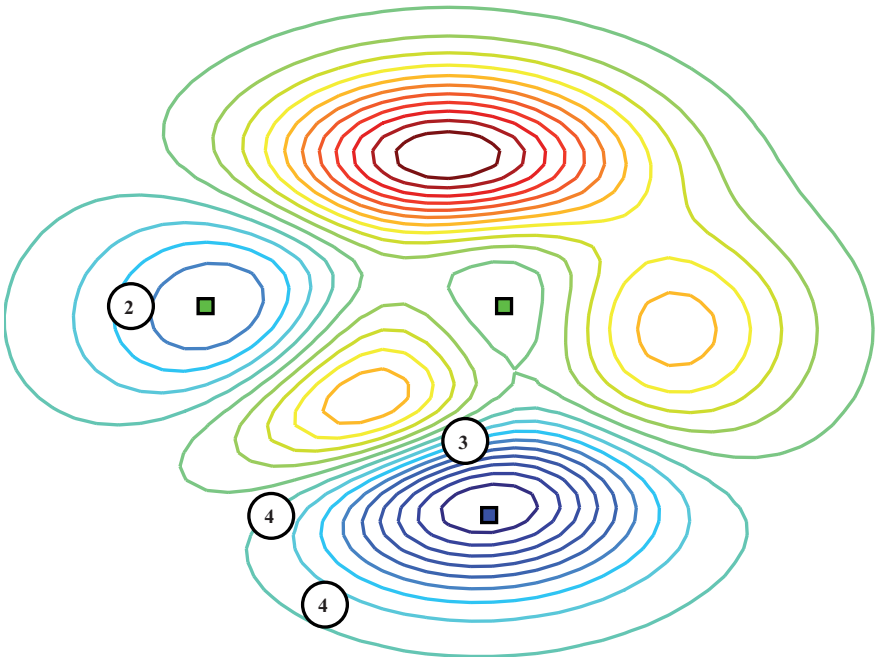


Figure 3.5: New *Refset*

the algorithm can either stop or perform a regeneration which usually consists in deleting the worst $b/2$ solutions in the *RefSet* in terms of quality and replacing them by diverse solutions. For this purpose, the *Diversification Generation Method* is used again to generate a set of diverse solutions (or we can simply use again the same set of diverse solutions used in the initialization of the method). Following the same diverse criterion as the one used in the first *RefSet* formation (see Section 3.2.2) those solutions maximizing their distance to the current *RefSet* solutions will be added to the *RefSet*. If the *RefSet* should be regenerated in a situation like that depicted in Figure 3.5, the 2 solutions to be deleted would be those with function value equal to 4 (since they are the b_2 worst solutions in terms of quality). The *Diversification Generation Method* would create new diverse solutions and those maximizing their distance with respect to the remaining solutions in the *RefSet* would enter it. Note that new solutions gain the *RefSet* membership one by one in a sequential way by maximizing their distance to the current solutions in the *RefSet*. In Figure 3.6 the 2 solutions in red would replace the deleted solutions.

3.2.6 Improvement method

As stated in Section 3.1, during the optimization procedure all the generated solutions are subjected to the *Improvement Method* which usually consists of a local search procedure to

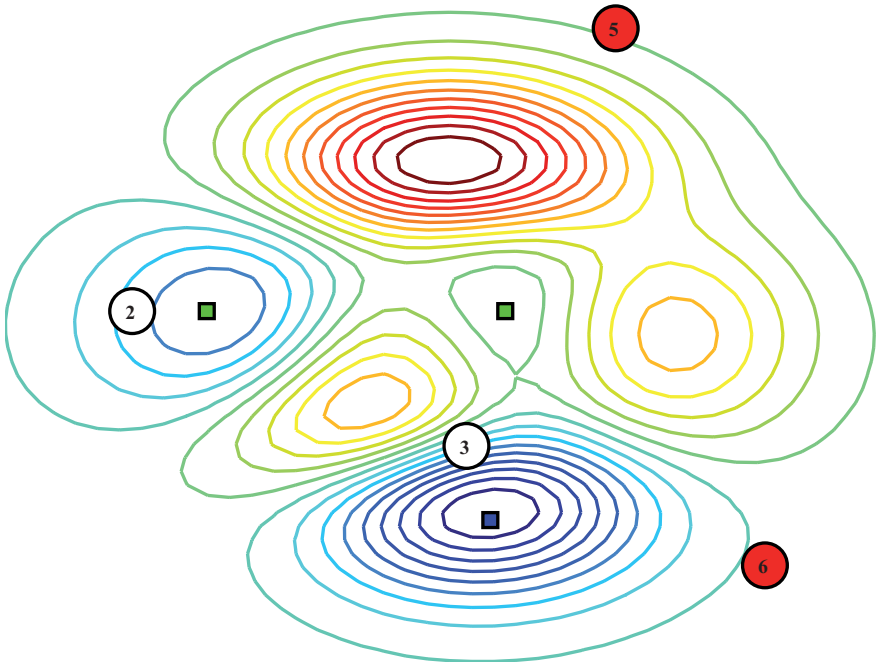


Figure 3.6: *Refset* regeneration

improve the quality of the solutions. In some applications where the local search provides poor results it might be suppressed or an alternative *Improvement Method* must be designed (such as another metaheuristic hybridized with scatter search). Here we will only illustrate how the *Improvement Method* usually works by applying it to two solutions (the best 2 solutions of our particular *RefSet*). In Figure 3.7 the application of the *Improvement Method* to two different solutions results in finding two optima, one of them being the global optimum.

Advanced scatter search implementations take into account different parameters, such as quality of the solutions and distances to found local minima in order to minimize the computational effort, and thus avoiding to perform local searches from initial solutions which are likely to provide already known local minima.

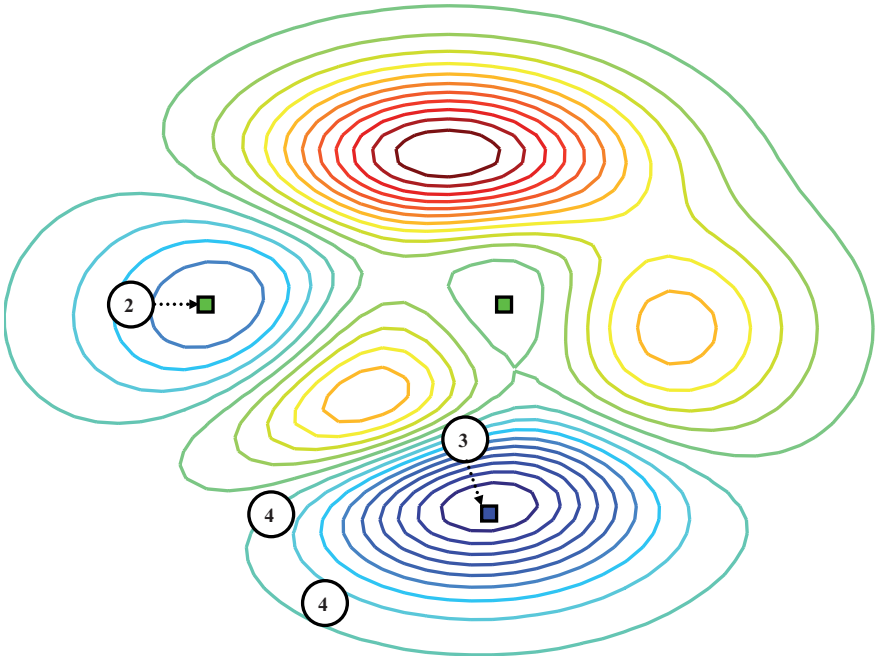


Figure 3.7: *Improvement Method* applied to 2 solutions in the *RefSet*

Part II

Methodology

Chapter 4

A scatter search heuristic for bioprocess optimization

4.1 Introduction

Many real world optimization problems in bioprocess engineering (and also in business or economics) are too complex to be given tractable mathematical formulations. Although we used mathematical notation in the formulations provided in Section 1.1, we are considering the general case in which there is no explicit expression of the objective function since it contains multiple nonlinearities, combinatorial relationships and uncertainties inaccessible to modeling except by resorting to more comprehensive tools like computer simulation. In the context of optimizing simulations, a "complex evaluation" refers to the execution of a simulation model (which can be extremely time-consuming).

Theoretically, the issue of identifying best values for a set of decision variables falls within the realm of optimization. Until quite recently, however, the methods available for finding optimal decisions have been unable to cope with the complexities and uncertainties posed by many real world problems of the form treated by simulation. The area of stochastic optimization has attempted to deal with some of these practical problems, but the modeling framework limits the range of problems that can be tackled with such technology. The complexities and uncertainties in these systems are the primary reason to often choose simulation as a basis for handling the decision problems associated with them. Advances in the field of metaheuristics have led to the creation of optimization engines that successfully guide a series

of complex evaluations with the goal of finding optimal values for the decision variables.

As stated in Chapter 1, most optimization problems in the chemical and bio-chemical industries are highly nonlinear in either the objective function or the constraints. Moreover, they show dynamic nature and can be formulated as nonlinear programming problems subject to differential-algebraic constraints. This set of constraints must be solved using specific mathematical techniques (e.g., initial value problem numerical methods) provided by the user or embodied in the objective function value to be optimized. Once this set of DAE's has been solved, and the objective function and additional constraints have been evaluated, the output information is used by the optimization procedure to drive the search and choose the new solutions to be evaluated (i.e., the new inputs for the DAE's solver) in an iterative way. Figure 4.1 shows the interaction of the optimization procedure with the external mathematical method to solve the set of differential-algebraic constraints of the dynamic model to be optimized.

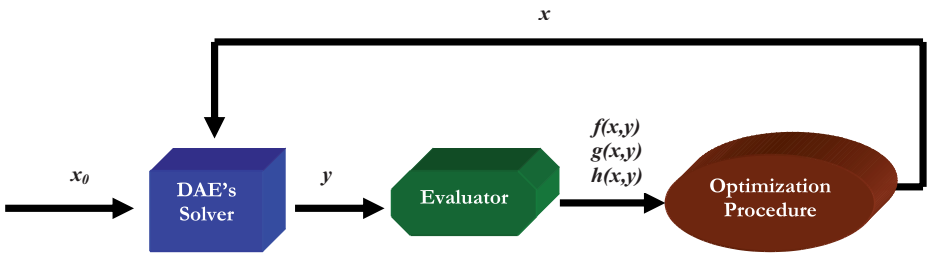


Figure 4.1: Interaction between the optimization procedure and the DAE's solver

In this context, optimization methods should be able to treat the optimization problems as "black-boxes". The disadvantage of "black-box" approaches is that the optimization procedure is generic and has no knowledge of the process employed to perform evaluations inside the box and therefore does not use any problem-specific information. The main advantage, on the other hand, is that the same optimizer can be applied to complex systems in many different settings. Therefore, although we have designed and tested our method in the process systems engineering environment, it can be directly applied to solve any kind of black-box optimization problems in other settings.

When classifying the global optimization algorithms in Section 1.2, it was explained that deterministic (or exact) algorithms work well only for small problems with some requirements (rarely met in real applications). Although they ensure the convergence to the global optimum, the computation time needed might be unaffordable. In contrast, stochastic (or

heuristic) methods can locate the global optimum or its vicinity in modest computation times for every type of problem. Many authors (e.g., see Banga et al. 2003c and references therein) have successfully applied stochastic global optimization methods to process engineering optimization.

In Chapter 2 we reviewed a kind of stochastic methods known as metaheuristics which arose to solve hard combinatorial optimization problems and were extended for solving continuous and mixed-integer problems in recent years. Many examples of application of metaheuristics to process engineering optimization were provided.

Current research in global optimization is highly devoted to metaheuristics. In this work, we propose a scatter search-based global optimization algorithm for continuous and mixed-integer problems which is efficient for solving optimization problems arising in bioprocess engineering. The motivation for choosing scatter search as a basis of our algorithm is that, in a recent review by Neumaier et al. (2005) comparing several GO solvers over a set of 1000 constrained GO problems, the scatter search-based algorithm *OQNLP* (see Section 6.1) obtained the best performance among the stochastic methods, solving the highest percentage of big problems.

4.2 Methodology

We have implemented our algorithm in Matlab under the name *SSm* (Scatter Search for Matlab). Our development goes beyond a simple exercise of applying scatter search to optimization problems in bioprocess engineering, but presents innovative mechanisms to obtain a good balance between intensification and diversification in a short-term search horizon. In the Applications part of this work, computational comparisons with the selected methods introduced in Section 6.1 over a set of bioprocess engineering optimization problems favor the proposed procedure.

4.2.1 *Diversification Generation Method*

SSm begins by generating an initial set S of m diverse vectors in the search space. Unlike other diversification strategies, *SSm* does not only generate vectors with their components uniformly distributed within the search space, but also drives the generation of values for each decision variable onto parts of the space where they have not appeared very often during the diversification process. For that, the method makes use of memory taking into account the

number of times that every decision variable appears in different parts of the search space. This is usually accomplished by dividing the range of each variable into p sub-ranges of equal size. Then, a solution is constructed in two steps. First, a sub-range is randomly selected. The probability of selecting a sub-range is inversely proportional to its frequency count (which keeps track of the number of times the sub-range has been selected). Second, a value is randomly chosen from the selected sub-range.

Initially, the range of every decision variable, x_i with $i \in [1, 2, ..., n]$ (being n the problem size) defined by its lower and upper bounds, xl_i and xu_i respectively, is divided in p sub-ranges of equal size, $(xu_i - xl_i)/p$. Therefore, the limits that define each sub-range $j \in [1, 2, ..., p]$ for the variable i are given by

- Lower bound:

$$lb_{ij} = xl_i + \frac{xu_i - xl_i}{p}(j-1) \tag{4.1}$$

- Upper bound:

$$ub_{ij} = xl_i + \frac{xu_i - xl_i}{p}j \tag{4.2}$$

Frequencies,f_{ij}, are defined as the number of times that the variable i is in the sub-range j along all the generated vectors.

To initialize all the frequencies to a value of 1, p vectors are first generated, each of them having all their variables randomly generated in the same sub-range using a uniform distribution (e.g., vector 1, x^1, has all its variables in sub-range 1, and every decision variable i is randomly generated using a uniform distribution within the bounds xl_i and $xl_i + \frac{(xu_i - xl_i)}{p}$). This first set of vectors forms the initial matrix of diverse vectors $S^{p \times n}$ that will be extended up to a size of m-by-n by adding new diverse vectors.

$$S = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^p \end{bmatrix} = \begin{pmatrix} x_1^1 & x_2^1 & \dots & x_n^1 \\ x_1^2 & x_2^2 & \dots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^p & x_2^p & \dots & x_n^p \end{pmatrix} \tag{4.3}$$

The initial matrix of frequencies is defined as:

$$f = \begin{pmatrix} f_{11} & f_{12} & \dots & f_{1p} \\ f_{21} & f_{22} & \dots & f_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ f_{n1} & f_{n2} & \dots & f_{np} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix} \tag{4.4}$$

New vectors will be generated using the following procedure: for each new vector x^{p+t} to be generated (with $t \in [1, 2, ..., m-p]$) the probability of having its decision variable i in the sub-range j is calculated as

$$prob_{i,j}^{p+t} = \frac{\frac{1}{f_{ij}}}{\sum_{k=1}^{p} \frac{1}{f_{ik}}} \tag{4.5}$$

Then, a uniformly distributed random number, rnd, in the interval $[0, 1]$ is generated. The next generated vector x^{p+t} will have its i-th component in the subrange $j = a$ for the first value of a that accomplishes

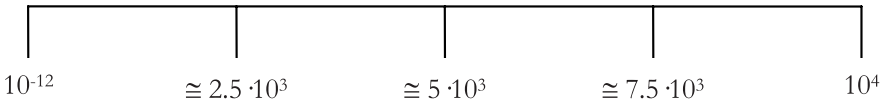
$$rnd \leq \sum_{j=1}^{a} prob_{i,j}^{p+t} \quad a = 1, 2, \ldots, p \tag{4.6}$$

Each component, x_i^{p+t}, will take a value randomly selected using an uniform distribution in the range $[lb_{ij}, ub_{ij}]$. Thus, for a new vector to be generated, the probability of having the variable i in the sub-range j is inversely proportional to the frequency of appearance of the variables i in this sub-range considering the already created vectors. Therefore, the method has to "remember" and update these frequencies to enhance diversity. As new vectors x^{p+t} are generated, they are added to the matrix S in rows until it becomes m-by-n dimensional. The starting set of points also includes the following three solutions: the first one in which all variables are set to the lower bound, the second one in which all variables are set to the upper bound, and the third one in which all variables are set to the midpoint between both bounds. This is the standard scatter search implementation of the *Diversification Generation Method* for non-linear problems and is used by different methods like *OptQuest* (Laguna and Martí, 2002). However, we have found that in some instances in which variables may have values in a huge range of positive values, a logarithmic distribution usually provides better results.

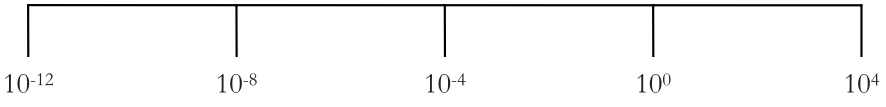
In the context of chemical and bio-process optimization, the selection of the lower bounds for the decision variables is usually quite straightforward because of their physical meaning (e.g., a temperature can never have a value lower than 0 Kelvin). However, the selection of the upper bounds is not so easy and they are often chosen as arbitrary large values to contain all the potential values for each variable. Therefore, it is expected that the optimal and good solutions may lie, in general, closer to the lower bounds than to the upper bounds. In this context, a uniform distribution for selecting diverse solutions within the bounds will

not generate many trial points with good values. In contrast, a logarithmic distribution will generate more trial vectors close to the lower bound, thus allowing the algorithm to be initialized with high quality members in the initial population, ensuring a faster convergence. Moreover, a logarithmic distribution is also helpful in the case of variables that can intrinsically have values in different orders of magnitude (as is the case of pre-exponential factors in kinetic equations) or with variables without physical meaning, for which selecting bounds is a difficult task. In order to obtain good initial values for these cases, an option for selecting variables in different orders of magnitude has been added in our implementation under the name *log_var*.

Figure 4.2 illustrates this situation. Consider a variable that takes values between 10^{-12} and 10^4. If we generate a starting set of points between those bounds using a uniform distribution, we will approximately obtain the same number of values in every interval shown in Figure 4.2(a). Alternatively, if we select the *log_var* option for this variable, its values will be randomly selected with equal probability across the sub-ranges depicted in Figure 4.2(b). With this option, the number of subintervals is automatically adjusted so that there are a maximum of two orders of magnitude between the limits of each interval (e.g., for a variable between 10^{-12} and 10^4, the number of subintervals would be 8), thus generating more solutions close to the lower bound.



(a) Values uniformly distributed within the bounds



(b) Values distributed within the different orders of magnitude

Figure 4.2: Intervals within a variable range

4.2.2 Building the *RefSet*

As described in Section 3.1, the *RefSet Update Method* is applied in two different steps of the algorithm: when building the initial *RefSet* from the set S of diverse solutions and when updating it after applying the *Solution Combination Method*.

For building the initial *RefSet*, after generating the set S of diverse solutions, two strategies may be chosen. In the first strategy (used by default), a subset of high quality and diverse points is selected as the *RefSet*. The initial *RefSet* is built selecting the best $b/2$ solutions from S as given by the evaluation-simulation process and then making more $b/2$ selections in order to maximize the minimum distance between the candidate solution and the solutions currently in *RefSet*.

The first step consists in evaluating all diverse vectors and select the $b/2$ best ones in terms of quality. For example, in a minimization problem, provided the diverse vectors are sorted according to their function values (the best one first), the initial selection is $\left[x^1, x^2, \ldots, x^{b/2}\right]^T$ such that

$$f(x^i) \leq f(x^j) \ \ \forall \ \ j > i \ , \ i \in [1, 2, \ldots, b/2 - 1] \ , \ j \in [2, 3, \ldots, b/2] \tag{4.7}$$

Vectors added to the *RefSet* are deleted from S. The current number of vectors present in the *RefSet* is computed as h. Therefore, in this stage $h = b/2$ (and the maximum possible value of h is b). We complete the *RefSet* with the remaining diverse vectors in S by maximizing the minimum Euclidean distance to the included vectors in the *RefSet*.

For every diverse vector in S, x^d, with $d \in [h+1, h+2, ..., m]$, Euclidean distances to all current *RefSet* vectors are computed. The minimum of these distances, d_{min}, is stored for each vector:

$$d_{min}(x^d) = min\{d(x^d, RefSet)\} \tag{4.8}$$

where $d(x^d, RefSet)$, represents a vector whose components are the Euclidean distances between vector x^d and all the vectors in the *RefSet*. Then, the vector $\mathbf{x}$ having the highest minimum distance will join the *RefSet*. Therefore, $RefSet = RefSet \cup \mathbf{x}$ such that

$$d_{min}(\mathbf{x}) = max\left(d_{min}(x^d)\right) \ \ \forall \ \ d = h+1, h+2, ..., m \tag{4.9}$$

and the value of h is increased one unit since a new vector has been added to the *RefSet*. This is repeated until the *RefSet* is filled with b vectors (i.e., $h = b$) so that $RefSet \in \mathbb{R}^{b \times n}$.

This criterion is applied in a sequential fashion. At each step we add to the *RefSet* the solution that maximizes $d_{min}(x^d)$, remove it from S, and then recalculate the Euclidean distances. Therefore, we add one solution at each step until the *RefSet* has been completed (i.e., we do it for $b/2$ steps).

This strategy requires $|S|$ simulations to identify the best $b/2$ solutions in terms of the objective function value. Unless we choose a low value for $|S|$, this can cause a waste of computational effort, especially in the case of time-consuming problems. We therefore propose an alternative strategy which does not take into account the quality of the diverse vectors. The initial *RefSet* is formed by 3 vectors: one having all the variables in their lower bounds, another one having all the variables in their upper bounds and the middle point between these two vectors. This initial $RefSet \in \mathbb{R}^{3\times n}$ is completed using the same distance criterion described in the first strategy until it is composed of b decision vectors.

Note that the first strategy involves a higher computational cost since all the diverse vectors have to be evaluated. However, this strategy ensures a better quality of the initial *RefSet* which can help to converge faster to the global solution. The second strategy does not involve any simulation prior to the optimization stage. We therefore have no information about the quality of these solutions and thus we expect the algorithm to converge more slowly. The first strategy combines quality and diversity in the initial *RefSet*, whereas the second one focuses only on diversity (and saves computational effort).

4.2.3 *Subset Generation* and *Solution Combination* methods

After the initial *RefSet* is built, its solutions are sorted according to their quality (i.e., the best solution is the first) and we apply the *Subset Generation Method*. Martí and Laguna (2003) stated that most of the quality solutions obtained by combination arise from sets of two solutions, thus, in our implementation, the *Subset Generation Method* consists of selecting all pairs of solutions in the *Refset* to combine them. To avoid repeating combinations with the same pair of solutions, we use a memory term which keeps track of the pairs previously combined.

The *Solution Combination Method* is a key element in scatter search implementations. This method is typically adapted to the problem context. Linear combinations of two solutions were suggested by Glover (1994) in the context of nonlinear optimization and are a generalization of the linear or arithmetical crossover also used in continuous and convex spaces (Michalewicz et al., 1994). Herrera et al. (2006) studied different types of combination procedures for scatter search applied to continuous problems. They concluded that the BLX-α algorithm (with $\alpha = 0.5$) is a suitable combination method for continuous scatter search. Laguna and Martí (2005) already used this idea and extended it to avoid generating

solutions in the same area by defining up to four different regions within and beyond the segments linking every pair of solutions. These authors changed the number of created solutions from each pair of solutions in the *RefSet* depending on the position of the latter in the sorted *RefSet*. Here we will use the same principles, but instead of performing linear combinations between solutions, we will perform a type of combination based on hyper-rectangles, which enhances the diversification.

These combinations are of the following four types, assuming that x' and x'' are the solutions to be combined and that x' is superior in quality to x'':

$$c_1 = x' - d_1 \tag{4.10}$$

$$c_2 = x' + d_2 \tag{4.11}$$

$$c_3 = x'' - d_3 \tag{4.12}$$

$$c_4 = x'' + d_4 \tag{4.13}$$

where $d_i = r_i \bullet (x" - x')/2$ with $i = 1, 2, 3$ or 4 depending on the number of solutions generated (see below); r_i is a vector of dimension n with all its components being uniformly distributed random numbers in the interval $[0, 1]$. The notation ($\bullet$) above indicates an entrywise product (i.e., the vectors are multiplied component by component), thus it is not a scalar product. The vector d_i has the following form:

$$d_i = \begin{pmatrix} d_{i,1} \\ d_{i,2} \\ \vdots \\ d_{i,n} \end{pmatrix}^T = \begin{pmatrix} \frac{r_{i,1}(x''_1 - x'_1)}{2} \\ \frac{r_{i,2}(x''_2 - x'_2)}{2} \\ \vdots \\ \frac{r_{i,n}(x''_n - x'_n)}{2} \end{pmatrix}^T \tag{4.14}$$

Note that if both solutions, x' and x'', belong to the first $b/2$ elements of the sorted *RefSet*, then 4 vectors are generated: one each type. If only x' belongs to the first $b/2$ elements of the sorted *RefSet*, then 3 vectors are generated (types 1, 2 and 4). Finally, if neither x'' nor x' belong to the first $b/2$ elements of the sorted *RefSet*, then 2 vectors are generated: one of type 2 and another one of type 1 or 3 (randomly chosen). Figure 4.3 illustrates the type of combinations and the regions in which new solutions are created.

These vectors generated by combination of the *RefSet* members will be named x^c with $c \in [1, 2, ..., nc]$, and form a matrix $C \in \mathbb{R}^{nc \times n}$ where nc is the total number of vectors generated by combination, which is not a fixed number. It may change in every iteration

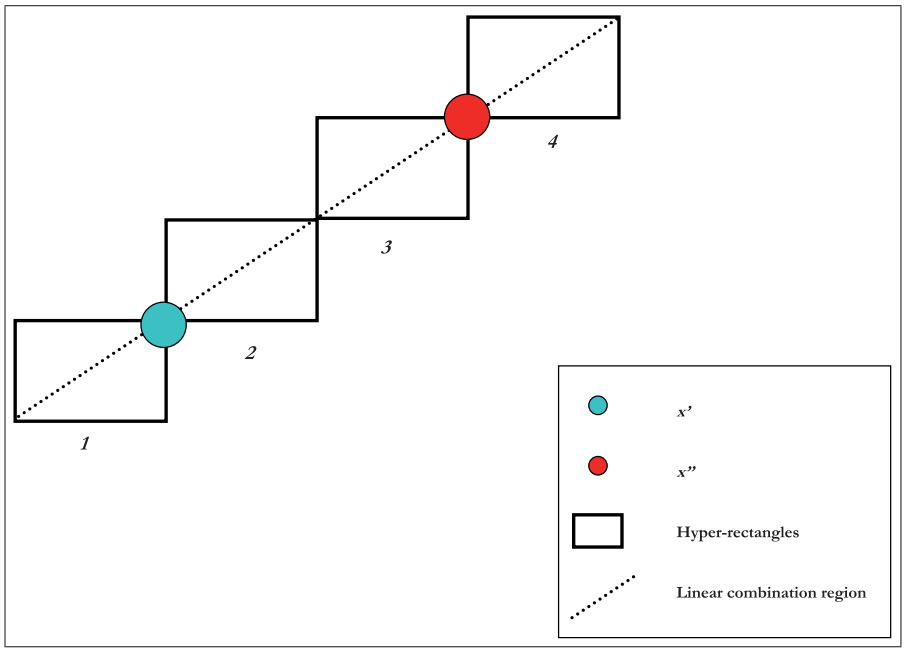


Figure 4.3: Combination method

depending on the number of combinations made among *RefSet* members (remember that the method avoids doing combinations with pairs of vectors already combined).

It must be noted that the *Solution Combination Method* does not have any checking mechanism to detect whether a new vector has any of its variables out of the bounds. Therefore, before evaluating these new vectors, a quick check is done by the algorithm: if any of the decision variables is out of the interval defined by its bounds, it is automatically adjusted to the value of the closest bound to it.

If the *RefSet* changes after the application of the *RefSet Update Method* described in Section 4.2.4, indicating that at least one new solution has been inserted in the *RefSet*, we apply again the *Solution Combination Method* to all the pairs in *RefSet* containing at least one new element. Otherwise, as in advanced scatter search designs, we resort to the rebuilding mechanism as described in Section 4.2.6.

4.2.4 Updating the *RefSet*

In its original design, the *Reference Set Update Method* indicates that the *RefSet* is updated by selecting a set of high-quality and diverse solutions from the union of the *RefSet* and the new combined solutions. In Martí and Laguna (2003), it is pointed out that the *RefSet* is usually updated considering the quality of the elements. However, we have empirically found that,

for continuous problems, this standard mechanism tends to create clusters of solutions, which results in an intermediate *RefSet* with very similar solutions which are unlikely to produce new good solutions by combination. This effect also appears in other metaheuristics applied to continuous problems and have been overcome in different ways (see for example Herrera and Lozano 2000 and Tfaili and Siarry 2008). We have added a distance filter to prevent similar solutions from becoming part of the *RefSet*. Specifically, we define a threshold value, *dth*, as a minimum Euclidean distance to be accomplished by every solution. This mechanism is illustrated in Figure 4.4: in a minimization problem, having 5 candidate solutions to form a *RefSet* of 4 members and a defined *dth*, we would start adding to the new *RefSet* the solution with the highest quality. We would add the rest of solutions to fill the *RefSet* regarding their quality and providing they comply with *dth*. In Figure 4.4, after adding the first two solutions (with function values equal to 1 and 3 respectively) we analyze the next candidate by quality (i.e., the solution with function value equal to 4). Since its distance to one of the solutions already in *RefSet* (i.e., the solution with function value equal to 3) is lower than the specified *dth*, it would not enter the *RefSet* and the procedure would continue analyzing the following candidate.

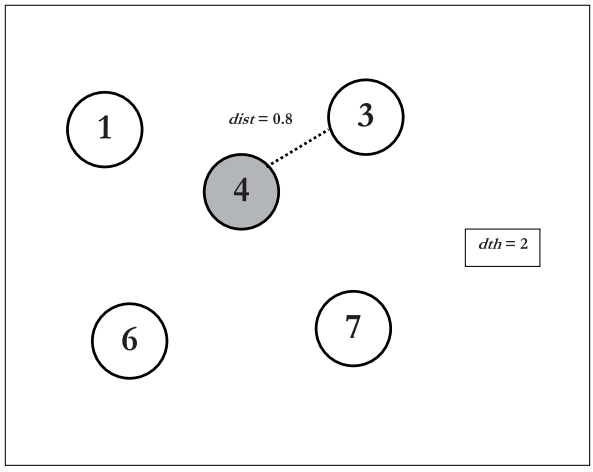


Figure 4.4: *RefSet* update with a threshold distance

This filter avoids clustering of the *RefSet* solutions thus preventing the search to prematurely converge to a sub-optimal solution. The price to be paid is to diminish the average quality of the *RefSet*, which may be a drawback for problems with a small budget of simulations.

The parameter *dth* is initialized as the minimum Euclidean distance among the members of the first *RefSet*. It increases or decreases its value dynamically depending on the search

history. If the best solution found does not improve in 2 consecutive iterations, *dth* decreases its value 10%. If we improve the best solution in 4 consecutive iterations, *dth* increases its value 10%. We have empirically found that, with this scheme, the *dth*-value is reduced in the last iterations, permitting the final refinement of the solutions.

A different criterion has been implemented for the cases in which the Euclidean distance criterion is inefficient. Indeed, we normalize Euclidean distances with respect to the bounds of the decision variables in order to have a similar contribution of each variable regardless their order of magnitude. If these bounds are not wisely chosen (e.g., the variables have no physical meaning or we simply have no idea of the practical range of them), this strategy can make the elements in the *RefSet* not to be as diverse as we wish. To avoid this, a second strategy based on differences in the decision variables can be chosen. According to this criterion, two solutions will be different if their relative difference in all variables is equal or greater than a value specified by the user.

We have included a second filter to prevent the method from being trapped in a region for a large number of iterations. In particular, if a solution is relatively far from the *RefSet* members but presents a very similar objective value to any of them (as it may happen in functions with flat landscape), we do not allow it to enter the *RefSet*. This prevents vectors in the same flat area from joining the *RefSet* at the same time. Provided the diversity criterion (defined by *dth*) is accomplished, the candidate vector z will join the *RefSet* only if

$$f(z) > f(x)(1+\varepsilon) \;\; \forall \;\; x \in RefSet \tag{4.15}$$

where ε is a small value defined by the user.

Figure 4.5 illustrates this situation in a minimization problem. Consider a solution x in the *RefSet* and a candidate solution z to enter it. Suppose that z verifies the distance filter according to *dth* and x has a slightly better value (say around 0.1% lower) than z. Then, instead of directly adding z to the *RefSet*, the quality filter considers that it may lie in the same flat area as x and forbids the action in order to"wait" for a more diverse solution.

In accordance with the problem's characteristics, the user adjusts this filter value, ε, for an optimal algorithm performance. The default value for this filter is relatively conservative, but it should be changed in problems in which we want to enhance diversity (for example when there are multiple local minima and the global optimum has a small basin of attraction). When relying on local search, the search may be more aggressive, whereas if no *Improvement*

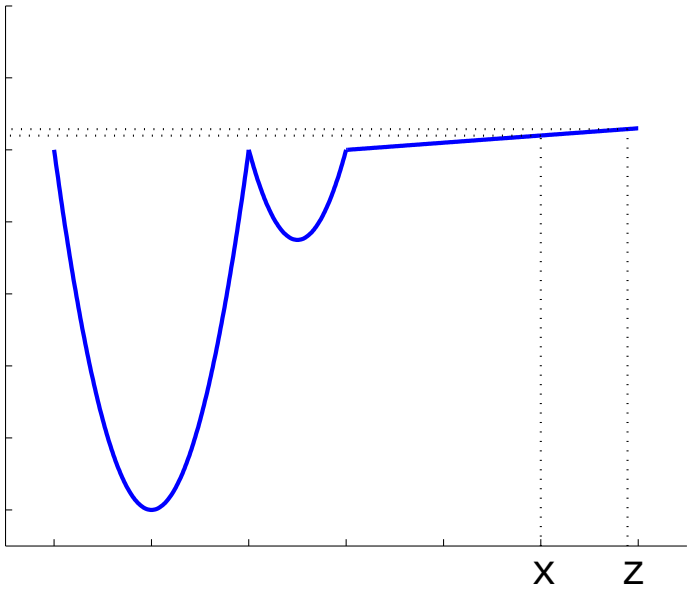


Figure 4.5: Two solutions in a flat zone of the objective function

Method is present, it is recommended that the default conservative value is used. In more advanced designs, this filter could be dynamic, being more relaxed at the beginning of the search in order to quickly locate the basin of attraction of the global minimum, and tighter at the end of the search to allow a specified tolerance in the solution. The evolution of this filter is not obvious and needs experimental work. As is shown in the Applications part, the algorithm performs very well with a constant value.

To summarize how the *RefSet Update Method* is working in our algorithm taking into account the filters described, Algorithm 14 shows a pseudocode of the procedure.

4.2.5 *Improvement Method*

The *Improvement Method* consists of a local search with the appropriate algorithm, using a carefully selected solution as the starting point. One of the advantages of implementing our optimization method in the Matlab environment is that we can easily apply any improvement method available in one of the many existing libraries. We have considered the following methods:

- *fmincon*: this is a local gradient-based method, implemented as part of the Matlab optimization Toolbox, which finds a local minimum of a constrained multivariable function by means of a SQP (Sequential Quadratic Programming) algorithm. The method uses

Algorithm 14 *SSm RefSet Update*

1: Create a set, *tempset*, with the *RefSet* and the generated solutions by combination: $tempset \in \mathbb{R}^{b+nc\times n} = RefSet \cup C$
2: Sort solutions in *tempset* by quality
3: Clear the *RefSet*: $RefSet = \emptyset$
4: Add the best solution in *tempset* to the *RefSet* and clear it from *tempset*
5: **while** $|RefSet| < b$ **do**
6: **for** $i = 1$ to $|tempset|$ **do**
7: $x_t = x^1_{tempset}$
8: Delete $x^1_{tempset}$ from *tempset*
9: **if** $\min\left\{d(x_t, x^i)\right\} > dth$ **and** $f(x_t) > f(x^i)(1+\varepsilon)\ \ \forall\ \ x^i \in RefSet$ **then**
10: $RefSet = RefSet \cup x_t$
11: **break for**
12: **end if**
13: **end for**
14: **end while**

numerical or, if available, analytical gradients.

- *solnp*: the SQP method by Ye (1987).

- *fsqp*: this algorithm is a SQP method for minimizing smooth objective functions subject to general smooth constraints. The successive iterates generated by the algorithm all satisfy the constraints (Panier and Tits, 1993).

- *ipopt*: Interior Point OPTimizer is a software package for large-scale nonlinear optimization. It is designed to find (local) solutions of nonlinear programs (Wächter and Biegler, 2006).

- *misqp*: the solver called Mixed-Integer Sequential Quadratic Programming (MISQP) is a SQP Trust-Region method, recently developed by Exler and Schittkowski (2006, 2007), which handles both continuous and integer variables.

- *n2fb*: this algorithm was specially designed for non-linear least squares problems by Dennis et al. (1981). The method is based on a combined approximation of a Gauss-Newton and quasi-Newton algorithm.

- *lsqnonlin*: this method is also designed for non-linear least squares problems. It can use different algorithms such as the interior-reflective Newton method (Coleman and Li, 1994), the Levenberg-Marquardt method with line search (Moré, 1978) or a Gauss-Newton method with line search (Dennis, 1977).

- *fminsearch*: this is a direct search method implemented in Matlab, that uses the simplex search method of Lagarias et al. (1999). It does not use numerical or analytic gradients. We have adapted the original code to handle nonlinear constraints.

- *NOMADm*: Nonlinear Optimization for Mixed variables And Derivatives-Matlab, abbreviated as NOMADm (see Abramson 2002), is a Matlab code that runs various Generalized Pattern Search (GPS) algorithms to solve nonlinear and mixed variable optimization problems. This solver is suitable when local gradient-based solvers do not perform well.

- *dhc*: the Dynamic Hill Climbing algorithm by de la Maza and Yuret (1994) is a direct search algorithm which explores every dimension of the search space using dynamic steps. Only the local phase of the algorithm has been implemented.

In a classical implementation of scatter search, the improvement method is applied to a large number of solutions (all the initial solutions in S and all the combined solutions from the *RefSet*). However, in applications related to chemical and bio-process engineering, we often face time-consuming evaluation problems (i.e., every function evaluation can consume several minutes) or complex topologies which can make the local search inefficient. This implies that the application of the *Improvement Method* should be restricted to a low number of promising solutions. This idea has also been used by other authors in memetic algorithms, assigning different probabilities to the individuals to be subject to a local search (see for example Lozano et al. 2004; Molina et al. 2005). It is expected that in the first iterations of the search process the solutions generated will be of a relatively poor quality. Therefore, we have implemented a parameter that determines the iteration number in which the *Improvement Method* is applied for the first time (i.e., defining a number of previous function evaluations before calling the *Improvement Method*). Then, once this is satisfied, both quality and diversity filters are applied. These filters were successfully applied in Ugray et al. (2005) and they do not allow the *Improvement Method* to be applied from a solution of a low quality (merit filter), or from a solution close to other from which the *Improvement Method* was applied in previous iterations (diversity filter). As documented by these authors, the filters significantly reduce the computational time with good results.

The merit filter ensures that no local search will be performed unless we do not find a better initial point than those found before. However, this filter is flexible since finding high

quality solutions might be a time-consuming task for some problems and calling the local search from other points may be useful. An initial threshold is defined in the first call to the local search (e.g., as the function value of the first initial point). If no good initial points are found after a pre-defined number of iterations, the filter is relaxed (i.e., points with worse objective function values can be chosen as initial points to perform the local search) by means of a relaxation parameter th_{factor}. The threshold is relaxed calculating a new threshold from the existing one according with:

$$th_{new} = th_{old} + th_{factor}(1 + |th_{old}|) \tag{4.16}$$

The distance filter computes the Euclidean distance between the local minima and the initial points used to locate them. This filter prevents the algorithm from doing local searches from initial points that might lead to already found local optima. It assumes hyper-spherical basins of attraction for the optima and defines the radius of a hyper-sphere as the Euclidean distance from the initial point used for the local search and the optimum found. In practice, basins of attraction are not hyper-spherical, thus the distance filter often needs to be relaxed. If no initial points being far enough from found optima are found after a number of iterations, this filter is relaxed by multiplying the radii by a parameter in the interval $[0, 1]$.

To avoid overlapping of the hyper-spheres defining the initial points and their respective local minima, a correction of this filter has been implemented. Figure 4.6 represents two local searches which lead to two different optima, activating the distance filters (dotted circles). To prevent other vectors leading to different optima from being discarded as initial points for the following local searches, a correction of the filters is applied (solid circles).

For two different local minima, $\mathbf{x_i}$ and $\mathbf{x_j}$ their respective radii defining the distance filters are r_i and r_j, which must satisfy:

$$r_i + r_j \leq d(i, j) \tag{4.17}$$

where $d(i, j)$ is the Euclidean distance between $\mathbf{x_i}$ and $\mathbf{x_j}$.

The distance filter should be relaxed or even deactivated when many local solutions are closely located, as it is the case of some problems in bioprocess engineering optimization.

An alternative strategy has been implemented for applying the *Improvement Method* in our algorithm: instead of using filters, a local search is performed every time that the algorithm finds a better solution than the best current one, using this new found best solution as

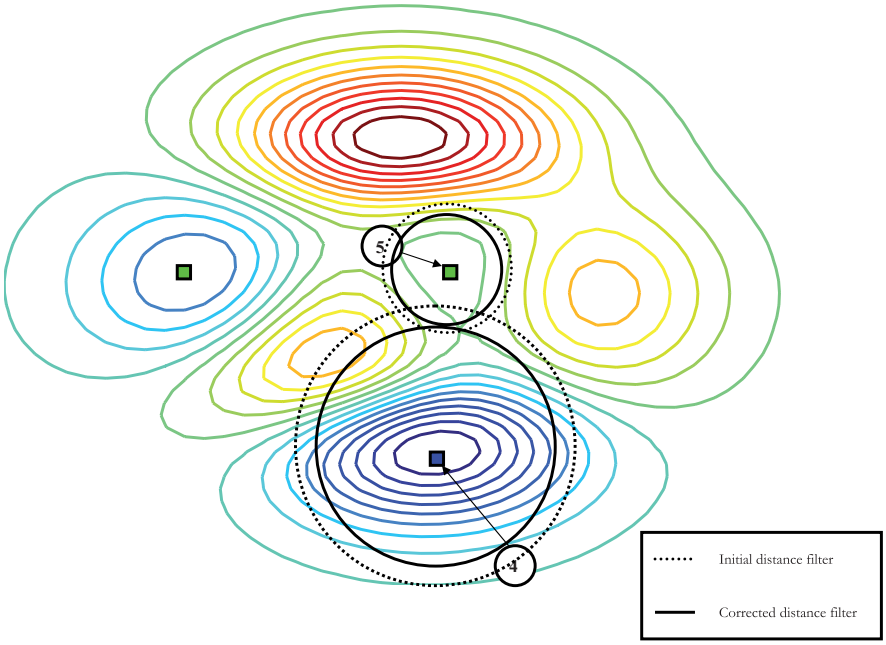


Figure 4.6: Correction of the distance filter overlap

initial point. To avoid performing many local searches from similar initial points, a minimum number of function evaluations between two local searches is fixed. This allows the algorithm to search more globally without spending computation time on intermediate local searches without improving the best solution. Algorithm 15 shows a pseudocode of this procedure.

Algorithm 15 Alternative *Improvement Method* strategy

```
Set n1, n2
Set neval = 0
Apply the Diversification Generation Method
Build the initial RefSet
Perform a local search from the best solution so far after n1 evaluations
xbest = Local solution obtained
Start the SSm main routine
while not end of the optimization do
  if a solution, x*, better than xbest found and neval ≥ n2 then
    Perform a local search from x*
    xbest = Local solution obtained
    neval = 0
  end if
  neval = neval + function evaluations of current iteration
end while
```

4.2.6 *RefSet* Rebuilding

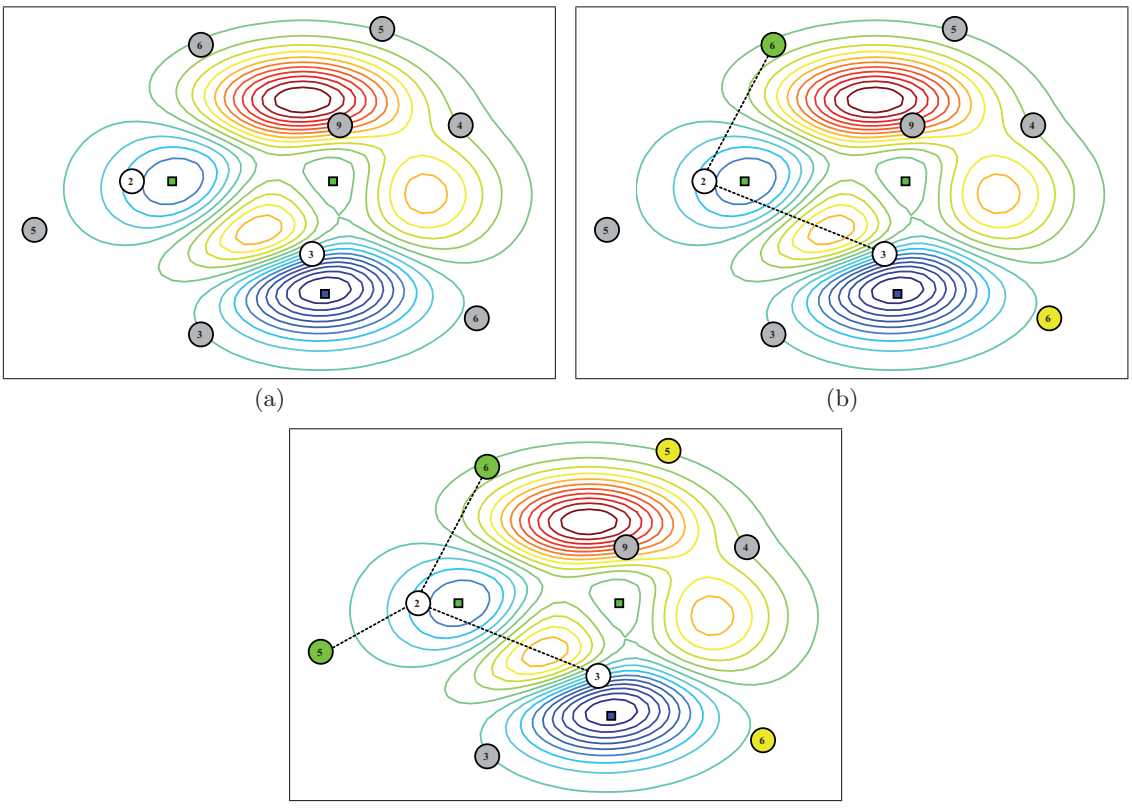
Rebuilding is a key operation associated with the *RefSet*. It implements a mechanism to partially rebuild the *RefSet* when none of the new trial solutions generated with the *Combi-*

nation Method, x^c, qualifies for addition to the *RefSet*. In advanced scatter search designs, the method is usually the same as that used to create the initial *RefSet*, in the sense that it uses the max-min distance criterion for selecting diverse solutions. Typically, the worst g vectors in *RefSet* (in terms of quality) are deleted. New diverse vectors are generated using the *Diversification Generation Method* and the *RefSet* is refilled according to the diversity criterion of maximizing Euclidean distances performed in the first *RefSet* formation. Normally, g is equal to $b/2$ but in aggressive implementations it can be set to $b-1$ (i.e., all the solution vectors in the *RefSet* except the best one are deleted).

We have modified the standard implementation of the rebuilding mechanism to incorporate the notion of orthogonality. Over a long-term horizon, the purpose of adding diverse solutions to the *RefSet* is to generate new search directions. It is therefore interesting not only to get scattered solutions in the search space, but also solutions that are able to create new search directions. Then, instead of selecting the solutions in S with the max-min distance, we select those with min-max cosine with the solutions already in the *RefSet*. Specifically, we choose the best element in *RefSet* as the center of gravity and in the first iteration apply the standard criterion to add the first diverse solution to the *RefSet*. Consider now the vector linking this new solution with the center of gravity. In subsequent iterations, instead of considering distances between the solutions in S and the *RefSet* members, we consider the vectors that the former define with the center of gravity and select the solution associated with the vector that minimizes the maximum value of the cosine among the vectors of the solutions already in the *RefSet*. In this strategy, the vectors refilling the *RefSet* are chosen to maximize the number of relative directions defined by them and the existing vectors in the *RefSet*.

Figure 4.7 illustrates both types of *RefSet Rebuilding*, the classical one, by distance, and our strategy, by direction. In Figure 4.7(a) two solutions of the *RefSet* (in white) have been kept, and the rest have been deleted. The *Diversification Generation Method* has created a set of diverse solutions (in grey) from which we will take two to refill the *RefSet*. The classical criterion of maximizing the Euclidean distance would select the yellow solution in Figure 4.7(b) as the next *RefSet* member. Our criterion selects the best solution remaining in the *RefSet* as a center of gravity and would generate all the vectors linking it to the rest of solutions in the *RefSet* (in this case just a vector since there are only two solutions). Among the candidate solutions generated by the *Diversification Generation Method*, we add to the

RefSet the one defining a vector with the center of gravity which is as orthogonal as possible to the rest of vectors defined by the latter and the rest of solutions in the *RefSet* (e.g., in Figure 4.7(b), we would add the green solution to the *RefSet*). The process is repeated with the following solutions to be included in the *RefSet*. In Figure 4.7(c), the classical criterion would again select the furthest solution from the current *RefSet* members (in yellow again). Our criterion would analyze all the possible vectors defined by linking the candidate solutions with the center of gravity and would select that solution defining a vector as orthogonal as possible to the rest of vectors defined by the center of gravity and the rest of solutions in the *RefSet* (in green again).



(a) (b) (c)

Figure 4.7: *RefSet* rebuilding by distance (yellow) and by direction (green)

In other words, after deleting the g worst solutions, the *RefSet* is $(b-g) \times n$ dimensional. Let $j = b - g$ be the number of existing vectors in the current *RefSet*. We introduce the new matrix $M^{(j-1)\times n}$ containing the vectors that define the segments formed by the best vector in *RefSet* (i.e., the center of gravity) and the rest of vectors in it. The $(k-1)th$ row of M is $x^1 - x^k$, being x^1 the center of gravity, and x^k $(k = 2, \ldots, j)$ the rest of the elements in it (note that the *RefSet* is sorted according to quality). For every diverse vector created with

the *Diversification Generation Method*, x^d with $d \in [1, 2, \ldots, m]$ in the regeneration phase, a vector Q^d of scalar products is also defined

$$Q^d = (x^1 - x^d)M^T \tag{4.18}$$

where x^1 is again the center of gravity and M^T is the transpose matrix of M. For every x^d, the maximum value of its corresponding vector Q^d is computed as $msp(x^d)$. The solution $y \in x^d$ will join the *RefSet* in the regeneration phase if

$$msp(y) = \min\{msp(x^d)\} \qquad \forall d \in [1, 2, \ldots, m] \tag{4.19}$$

At this stage, the value of j is increased one unit, the value of m is decreased one unit and the process continues until $j = b$. The application of this strategy results in a maximum diversity in search directions on the regenerated *RefSet*. After every time the *RefSet Rebuilding* is carried out, the center of gravity is allowed to be combined again with all the rest of *RefSet* solutions, regardless of the fact that it was previously combined with any of them. This has similarities with the *aspiration criterion* of tabu search, in which a forbidden movement is allowed if a predefined condition is met.

4.2.7 Intensification

The inclusion of the distance filter in the *RefSet Update Method* (Section 4.2.4) could be too restrictive if the parameter *dth* takes relatively large values, (or if the tolerance chosen by the user when using the other strategy is too high), thus rejecting too many solutions to become part of the *RefSet*. Instead of keeping this parameter under low values to prevent this effect, we have experimentally found that if we store the rejected solutions with good values in a secondary reference set, $RefSet_2$, we can treat them differently from the other solutions in the *RefSet*. $RefSet_2$ stores the solutions that do not qualify to enter into the *RefSet* and present a value close to the value of the best solution found (specifically, better than the second best solution in the *RefSet*). During the *Solution Combination Method*, we combine the best solution in the *RefSet* with all the solutions in $RefSet_2$ and add all the resulting solutions to the *RefSet Update Method* phase. This intensification mechanism is performed every $Intens_{freq}$ iterations.

Figure 4.8 illustrates those combinations in an example with four solutions in the *RefSet* (white and red circles) and two solutions in $RefSet_2$ (grey circles). The blue square represents

the global optimum. In this example this intensification strategy makes the process converge faster since the combination of solutions in $RefSet_2$ (in grey) with the best in *RefSet* (red circle) generates solutions which are very close to the global optimum of the function.

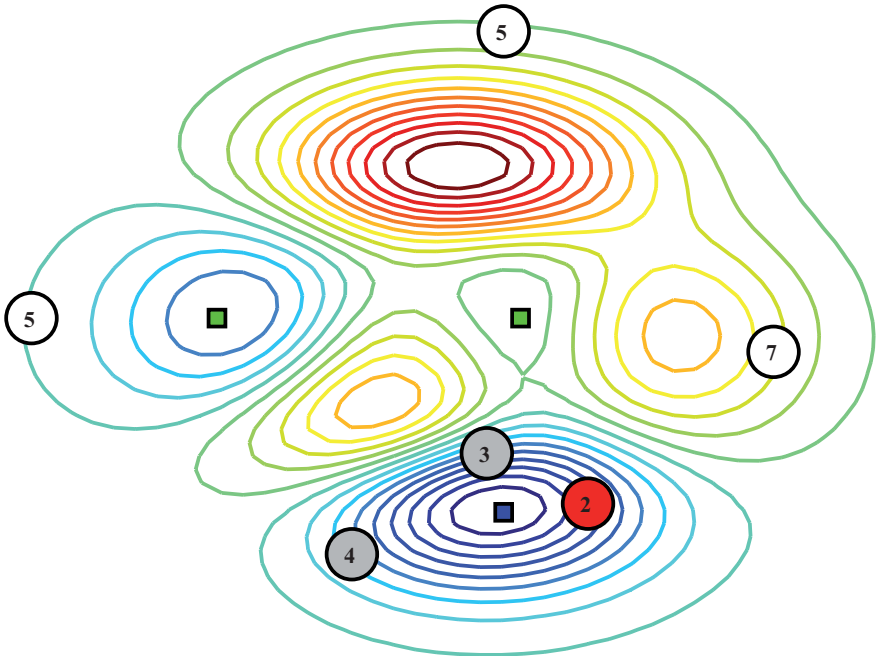


Figure 4.8: Intensification strategy

4.2.8 The *go beyond* strategy

Another advanced strategy to enhance the intensification of the search has been implemented in our algorithm. It has been named *go beyond* strategy and consists of exploiting promising directions. When performing the *Solution Combination Method* every generated vector is compared with its *parent*. If a new vector outperforms its parent in terms of quality, a new non-convex solution in the direction defined by the child and its parent is created. The child becomes the new parent and the new generated solution is the new child. If the improvement continues, we might be in a very promising area, thus the area of generation of new solutions is increased. This procedure is limited to the first $b/2$ elements of the *RefSet*.

A straightforward question arises from the last paragraph: *how do we identify the parent of a generated solution*? As explained in Section 4.2.3, the new solutions are created in hyper-rectangles defined by the pair of solutions combined (see Figure 4.3). The parent of a solution will be the *RefSet* solution lying in one of the vertices of the hyper-rectangle in which it has been created. Figure 4.9 depicts how the *go beyond* strategy works: from a pair

of *RefSet* solutions (in red), some new solutions are generated in the corresponding hyper-rectangles. The pink solution is the child whose parent is the *RefSet* solution in the vertex of its hyper-rectangle. Since the child outperforms the parent in quality, a new hyper-rectangle (in yellow) is defined by the distance between the parent and the child. A new solution (in orange) is created in this hyper-rectangle. This new solution becomes the child and the old child (i.e., the pink circle) becomes the parent. Since the new child (orange) outperforms again its parent (pink), the process is repeated, but the size of the new hyper-rectangle created (in green) is doubled because there has been improvement during two consecutive children generations.

As we can see in Figure 4.9, the new hyper-rectangle contains the global optimum, thus this strategy may locate it in a lower number of iterations than a scatter search without it. Algorithm 16 shows a pseudocode of the *go beyond* strategy procedure.

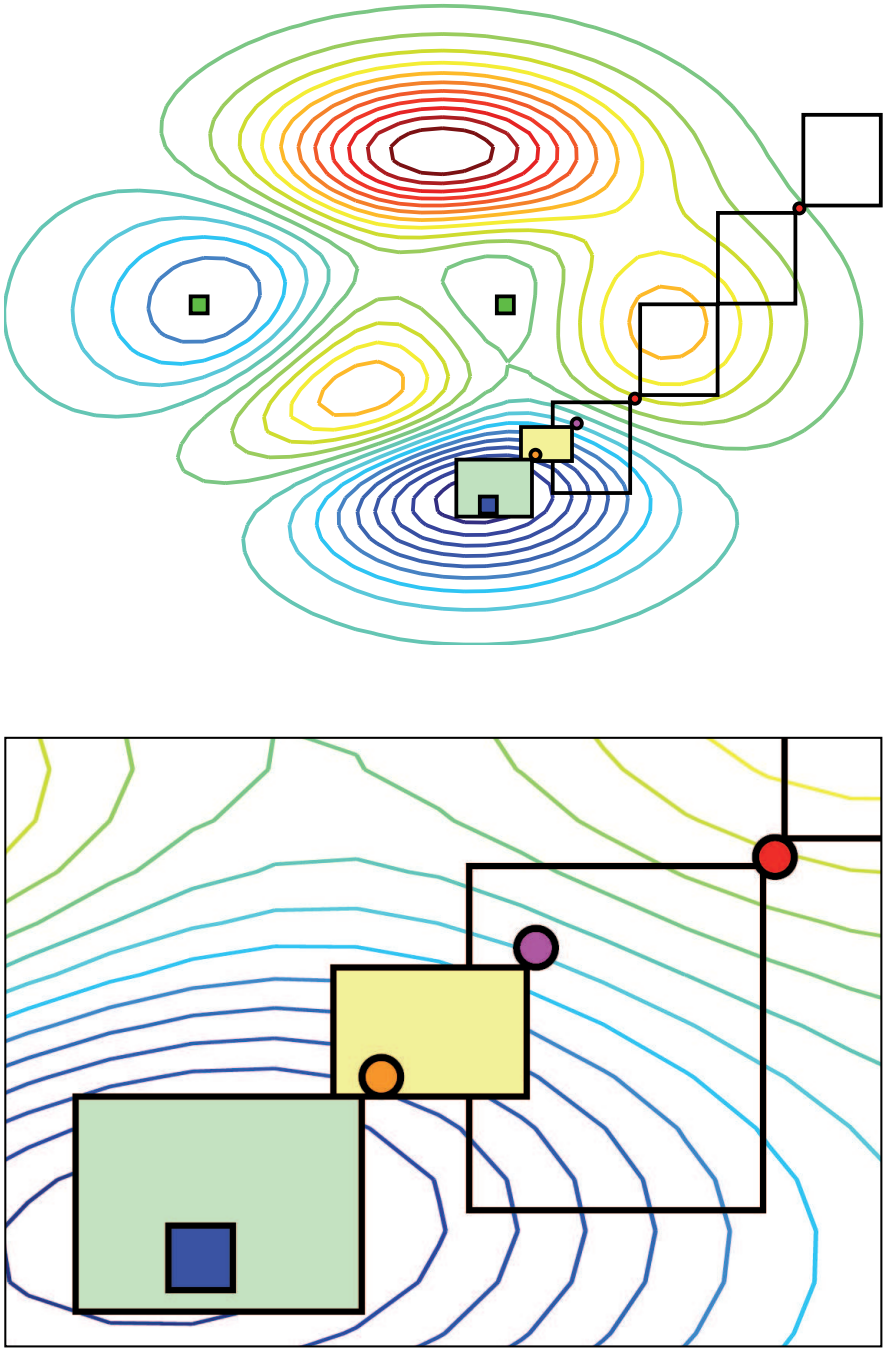
Algorithm 16 *go beyond* strategy

Call the *Solution Combination Method*
Identify children, $x_{children}$, outperforming their parents, x_{parent}
for i=1 to $|x_{children}|$ **do**
  $x_{ch} = x^i_{children}$
  $x_{pr} = x^i_{parent}$
  $improvement = 1$
  $denom = 1$
  while $f(x_{ch}) < f(x_{pr})$ **do**
    Create a new solution, x_{child_new}, in the rectangle defined by $[x_{ch} - \frac{x_{pr}-x_{ch}}{denom}, x_{ch}]$
    $x_{pr} = x_{ch}$
    $x_{ch} = x_{child\_new}$
    $improvement = improvement + 1$
    if $improvement = 2$ **then**
      $denom = denom/2$
      $improvement = 0$
    end if
  end while
end for

Even if the *go beyond* strategy has been mainly designed to enhance the intensification, the fact that the hyper-rectangles areas are increased if the new solutions improve the old ones during at least two consecutive iterations may make the search be more diverse, exploring regions where different minima can be found.

4.2.9 Constraints handling

Constraint handling of stochastic optimization methods has been a subject of research since these algorithms arose. Many different techniques have been proposed to handle problems



(b) (Zoom)

Figure 4.9: *go beyond* strategy

with constraints (e.g., see reviews of Michalewicz 1996; Coello Coello 2002; Yeniay 2005). In our algorithm, we have implemented a simple strategy consisting of a static penalty function. The objective function evaluated by the algorithm has the following form:

$$C(\mathbf{x}) = f(\mathbf{x}) + w \cdot \max\left\{\max\{viol(\mathbf{h}), viol(\mathbf{g})\}\right\} \tag{4.20}$$

where $\mathbf{x}$ is the solution being evaluated, $f(\mathbf{x})$ is the original objective function value, $\mathbf{h}$ is the set of equality constraints and $\mathbf{g}$ is the set of inequality constraints. w is a penalization parameter, which is constant during the optimization procedure (and usually has a high positive value). We use the $L-\infty$ norm of the constraints set to penalize the original objective function. Other penalty function approaches usually use the $L-1$ (exact penalization) or the $L-2$ (quadratic penalization). More sophisticated strategies might be implemented, although we strongly rely on the local solvers used by our algorithm to achieve feasible optimal points.

4.2.10 Integer variables handling

Many problems in the chemical and biotechnological industry, such as process design, process synthesis and multi-component blended-flow problems, lead to mixed integer nonlinear models (Adjiman et al., 1997, 2000; Kallrath, 2000, 2005).

Although our algorithm is mainly designed for continuous problems, a rounding operator has been implemented for handling integer and binary variables. Glover et al. (2000b) introduced an operator in scatter search to generate MIP solutions. Here we will use the rounding operator described by Ugray et al. (2005). The *Solution Combination Method* used by our algorithm does not take into account whether a decision variable has been declared as an integer variable or not. Thus, a rounding method has to be considered for this kind of variables before evaluating new solutions. For a decision variable x_i with $i \in [1, 2, ..., n]$, we transform it to its closest allowed integer y_i value by:

$$y_i = lb_i + \left[0.5 + \frac{x_i - lb_i}{st_i}\right] st_i \tag{4.21}$$

where lb_i is the lower bound for the decision variable i, and st_i is the step between two consecutive integer values (usually 1). If the calculated y_i is lower than the upper bound for that decision variable, ub_i, then y_i is accepted as the rounded value. Otherwise, we make $y_i = ub_i$.

4.2.11 Stopping criterion

The stopping criterion of our algorithm is taken as a combination of three conditions:

- Maximum number of evaluations exceeded.

- Maximum computational time exceeded.
- Specified value of the cost function reached.

By default, the algorithm will stop when any of these conditions is satisfied. Since the *Improvement Method* is selectively applied, when the scatter search algorithm is over, before abandoning the search, we apply the *Improvement Method* to the best solution found so far, just to be sure that it is not skipped, or simply to refine the best solution. In this final local search, the stopping tolerance assigned to the local solver is tighter than in the rest of the local searches performed along the optimization procedure.

In Appendix A, some documentation about the Matlab implementation of our algorithm, *SSm*, such as problem settings, options and application examples, is provided. Besides, two extra tools included in the same toolbox are also documented: *ssm_multistart*, for performing multistart local searches with the local solvers implemented in *SSm*, and *ssm_test*, to perform many optimizations over a set of benchmark problems using the same set of parameters or to test a single problem with different combinations of parameter values.

4.3 Application to benchmark problems

As a first test of our algorithm's performance, it has been applied to a set of well known unconstrained and constrained global optimization problems that have usually been used as benchmark problems in the literature for testing optimization software. The mathematical equations of all these test problems are listed in Appendix B. Additionally, to check its applicability to mixed-integer optimization, a set of this type of problems from the process engineering area has also been considered. For every problem tested in this section, the local solver chosen was *misqp*.

4.3.1 Unconstrained problems

We have tested our algorithm over 40 unconstrained problem of different dimensions. Table 4.1 provides information about all these problems.

Following the same procedure as in Laguna and Martí (2005), we have defined an optimality gap as:

$$GAP = |f(x) - f(x^*)| \tag{4.22}$$

Number of variables	Problem Number	Problem Name	x^*	$f(x^*)$
2	1	Branin	(9.42478, 2.475)[a]	0.397887
	2	B2	(0, 0)	0
	3	Easom	(π, π)	-1
	4	Goldstein and Price	(0, -1)	3
	5	Shubert	(-7.7083, -7.0835)[a]	-186.7309
	6	Beale	(3, 0.5)	0
	7	Booth	(1, 3)	0
	8	Matyas	(0, 0)	0
	9	SixHumpCamelback	(0.089840, -0.712659)[a]	-1.03163
	10	Schwefel(2)	(420.9687, 420.9687)	-837.9658
	11	Rosenbrock(2)	(1, 1)	0
	12	Zakharov(2)	(0, 0)	0
3	13	De Joung	(0, 0, 0)	0
	14	Hartmann(3,4)	(0.114614, 0.555649, 0.852547)	-3.86278
4	15	Colville	(1, 1, 1, 1)	0
	16	Shekel(5)	(4, 4, 4, 4)	-10.1532
	17	Shekel(7)	(4, 4, 4, 4)	-10.4029
	18	Shekel(10)	(4, 4, 4, 4)	-10.5364
	19	Perm(4,0.5)	(1, 2, 3, 4)	0
	20	Perm0(4,10)	(1, 1/2, 1/3, 1/4)	0
	21	Powersum	(1, 2, 2, 3)	0
6	22	Hartmann(6,4)	(0.20169, 0.150011, 0.47687, 0.275332, 0.311652, 0.6573)	-3.32237
	23	Schwefel(6)	(420.9687,..., 420.9687)	-2513.897
	24	Trid(6)	$x_i = i * (7 - i)$	-50
10	25	Trid(10)	$x_i = i * (11 - i)$	-210
	26	Rastrigin(10)	(0,..., 0)	0
	27	Griewank(10)	(0,..., 0)	0
	28	Sum Squares(10)	(0,..., 0)	0
	29	Rosenbrock(10)	(1,..., 1)	0
	30	Zakharov(10)	(0,..., 0)	0
20	31	Rastrigin(20)	(0,..., 0)	0
	32	Griewank(20)	(0,..., 0)	0
	33	Sum Squares(20)	(0,..., 0)	0
	34	Rosenbrock(20)	(1,..., 1)	0
	35	Zakharov(20)	(0,..., 0)	0
>20	36	Powell(24)	(3, -1, 0, 1, 3,..., 3, -1, 0, 1)	0
	37	Dixon and Price(25)	$x_i = 2^{-\frac{z-1}{z}}$, $z = 2^{i-1}$	0
	38	Levy(30)	(1,..., 1)	0
	39	Sphere(30)	(0,..., 0)	0
	40	Ackley(30)	(0,..., 0)	0

[a]This is one of several multiple optimal solutions.

Table 4.1: Unconstrained test problems

where x is a heuristic solution and x^* is the optimal solution. We say that a heuristic solution is satisfactory if:

$$GAP \leq \begin{cases} \varepsilon & \text{if} \quad f(x^*) = 0 \\ \varepsilon \, |f(x^*)| & \text{if} \quad f(x^*) \neq 0 \end{cases} \tag{4.23}$$

We set $\varepsilon = 0.0001$. For each test function we perform 30 independent runs with a

maximum number of function evaluations of 10^6. In any case, the optimization stops before achieving the maximum number of function evaluations if a satisfactory heuristic solution is found. The results obtained for this set of unconstrained problems are shown in Table 4.2.

Problem Number	$f(x^*)$	Results with *SSm*				
		Best	**Mean**	**Worst**	**% Success**	**Mean Evaluations**
1	0.397887	0.397887	0.397889	0.397896	100	236
2	0	1.10693e-009	5.11987e-006	7.90261e-005	100	3366
3	-1	-1	-9.99994e-001	-9.99935e-001	100	3949
4	3	3	3	3.00001	100	258
5	-186.7309	-186.7309	-186.7309	-186.7309	100	300
6	0	1.87494e-008	3.60511e-006	4.08401e-005	100	247
7	0	7.57705e-012	1.76946e-006	8.62196e-006	100	245
8	0	2.45572e-008	1.70451e-005	7.61722e-005	100	273
9	-1.03163	-1.03163	-1.03163	-1.03162	100	246
10	-837.9658	-837.9658	-837.9623	-837.9326	100	683
11	0	1.44532e-008	2.69161e-006	1.05570e-005	100	278
12	0	3.64625e-012	1.60548e-006	7.99405e-006	100	256
13	0	1.09026e-018	1.08278e-006	7.57465e-006	100	340
14	-3.86278	-3.86278	-3.86277	-3.86250	100	367
15	0	2.03938e-007	3.03223e-006	9.55533e-006	100	608
16	-10.1532	-10.1532	-10.1532	-10.1532	100	586
17	-10.4029	-10.4029	-10.4029	-10.4029	100	649
18	-10.5364	-10.5364	-10.5364	-10.5364	100	649
19	0	2.46504e-006	5.92432e-003	1.30655e-001	37	635689
20	0	7.20106e-007	2.63873e-005	9.61754e-005	100	4097
21	0	6.55970e-007	2.18288e-005	9.71297e-005	100	6118
22	-3.32237	-3.32237	-3.31044	-3.20316	90	132328
23	-2513.897	-2513.897	-2458.564	-2277.021	60	408437
24	-50	-50	-50	-50	100	752
25	-210	-210	-210	-210	100	1223
26	0	2.85055e-006	2.45423	6.96471	3	969903
26[a]	0	2.57876e-005	4.01953e-005	5.11601e-005	100	2943
27	0	1.60214e-006	1.16388e-005	2.80477e-005	100	18434
28	0	9.77524e-007	3.83435e-006	7.64475e-006	100	1381
29	0	1.06683e-006	1.06683e-006	1.06683e-006	100	2089
30	0	2.24968e-007	3.020977e-006	7.94442e-006	100	1248
31	0	3.97984	7.528523	15.9193	0	1000140
31[a]	0	4.51242e-005	7.08546e-005	9.79561e-005	100	10432
32	0	6.44905e-006	2.04312e-005	4.08787e-005	100	2753
33	0	1.00472e-006	3.88733e-006	6.90130e-006	100	2934
34	0	5.22079e-007	5.22079e-007	5.22079e-007	100	5573
35	0	2.31524e-008	3.49472e-006	9.00947e-006	100	2466
36	0	9.72116e-006	2.67765e-005	5.45857e-005	100	3772
37	0	1.72965e-006	5.55556e-001	6.66667e-001	17	835211
38	0	1.09245e-007	5.71577e-006	4.01198e-005	100	84167
39	0	5.47119e-015	5.647805e-007	1.50361e-006	100	3341
40	0	8.23867e-006	1.215999e-005	1.81072e-005	100	172538

[a]Solution found using a direct local search (*NOMADm*).

Table 4.2: Unconstrained test problems results

Our algorithm was able to find satisfactory solutions for most of the problems with a high

probability. However, it was not able to find good solutions for problem 31 in none of the 30 runs. For problem 26 (which is actually the same as problem 31: the *Rastrigin* function) only one run out of 30 was successful. By changing the type of local search we were able to solve these two problems. Instead of a gradient-based algorithm as *misqp*, a direct search method (*NOMADm*) was used. This algorithm may be able to overcome small local minima in the surroundings of the global optimum, thus increasing the probability of finding satisfactory solutions.

4.3.2 Constrained problems

The next set of problems used for testing our algorithm's performance is a collection of constrained problems usually used for testing new optimization software. Table 4.3 provides information about all these problems.

Problem Name	**Number of variables**	**Number of inequality constraints**	**Number of equality constraints**	$f(x^*)$
g01	13	9	0	-15
g02	20	2	0	-0.803619
g03	10	0	1	-1
g04	5	6	0	-30665.54
g05	4	2	3	5126.489
g06	2	2	0	-6961.814
g07	10	8	0	24.30621
g08	2	2	0	-0.095825
g09	7	4	0	680.6301
g10	8	6	0	7049.25
g11	2	0	1	0.75
g12	3	729	0	-1
g13	5	0	3	0.0539498

Table 4.3: Constrained test problems

As for unconstrained problems, we perform 30 independent runs for each problems with the default parameter values of our algorithm. In this case, we fix a maximum number function evaluations of 100000 in order to compare our results with those presented by Landa Becerra and Coello-Coello (2006). In their study, these authors compare the results obtained with a *Cultured Differential Evolution, CDE*, applied over the same set of problems. They outperform other optimization algorithms and carry out 30 independent runs per problem with a limit of 10^5 function evaluations each run. Table 4.4 shows our results compared with those reported by these authors.

Results obtained by our algorithm are competitive compared with those obtained by

Problem Name	$f(x^*)$	Results with *SSm*		Results with *CDE*	
		Best	**Mean**	**Best**	**Mean**
g01	-15	-15.00001	-14.66668	-15.00000	-15.00000
g02	-0.803619	-0.794662	-0.699783	-0.803619	-0.724886
g03	-1	-1.000049	-1.000034	-0.995413	-0.788635
g04	-30665.54	-30665.55	-30665.54	-30665.54	-30665.54
g05	5126.489	5126.498	5126.498	5126.571	5207.411
g06	-6961.814	-6961.821	-6961.815	-6961.814	-6961.814
g07	24.30621	24.30621	24.30621	24.30621	24.30621
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
g09	680.6301	680.6300	680.6300	680.6301	680.6301
g10	7049.25	7049.25	7049.25	7049.25	7049.25
g11	0.75	0.749990	0.749991	0.749900	0.757995
g12	-1	-1.000000	-1.000000	-1.000000	-1.000000
g13	0.0539498	0.0539495	0.143760	0.056180	0.288324

Table 4.4: Results for constrained problems, comparing with *CDE*

Landa Becerra and Coello-Coello (2006), which, at the same time, outperformed other state-of-the-art algorithms. Except in *g02* and the mean value of *g01*, our results are the same quality or even better (see *g05* and specially *g13*) than those reported by these authors, which indicates that our algorithm is competitive for constrained problems. Note that in some problems (e.g., *g03* and *g06*) our algorithm reports better values than the global optimum. This is because we allow by default a maximum constraint violation of 10^{-5} in the global phase. Besides, the local solver has its own tolerance which may allow a slight violation of constraints too.

4.3.3 Mixed-integer problems

To finish our algorithm's testing we have selected a set of constrained mixed-integer optimization problems arising from chemical engineering. The models used are written in AMPL code and can be found in Sven Leyffer's web page[1]. Table 4.5 shows information about this set of problems. Since, in some cases, the mathematical models describing the problems are quite large, they are not included here. We instead provide the original reference for each problem, where the model equations can be found.

Problems 1-5 names use the same notation as in Exler and Schittkowski (2006). Problems 6-13 names were taken from Leyffer (2001) and from the same author's web page. For this set of mixed-integer problem we followed the same procedure as in Section 4.3.1. The same optimality *GAP* is defined and 30 independent runs with a maximum number of 10^6 function evaluations were fixed. The results obtained for this set of mixed-integer problems are shown

[1] http://www-unix.mcs.anl.gov/∼leyffer/macminlp/

Problem Number	**Problem Name**	**Ref.**	ncv	niv	nbv	$f(x^*)$
1	mitp4	Asaadi (1973)	1	3	0	-40.957
2	mitp6	Asaadi (1973)	3	4	0	694.9
3	mitp8	Asaadi (1973)	4	6	0	37.219
4	mitp47	Kocis and Grossmann (1988)	2	0	3	7.6672
5	mitp49	Yuan et al. (1988)	3	0	4	4.5796
6	windfac	Michna (2000)	11	3	0	0.254487
7	synthes1	Duran and Grossmann (1986)	5	0	3	6.00976
8	synthes2	Duran and Grossmann (1986)	6	0	5	73.0353
9	synthes3	Duran and Grossmann (1986)	9	0	8	68.0097
10	optprloc	Duran and Grossmann (1986)	5	0	25	-8.06414
11	trimloss2	Harjunkoski et al. (1998)	6	0	31	5.3
12	batch	Kocis and Grossmann (1988)	22	0	24	285507
13	spring	Sandgren (1990)	5	1	11	0.846246

ncv = Number of continuous variables
niv = Number of integer variables
nbv = Number of binary variables

Table 4.5: Mixed-integer test problems

in Table 4.6.

As shown in Table 4.6, our algorithm is able to solve this set of mixed-integer benchmark problems. Due to some errors in the dynamic library function calling problem 11, only 20 optimizations instead of 30 were performed for it. Problem 13 was the hardest to be solved. Increasing the frequency of the local search call (i.e., making the search more aggressive), we achieved a higher percentage of success. Note that in problems 6-8 the best value obtained by our algorithm is better than the global solution. This is caused again by the default small constraint violation allowed by our algorithm.

With this set of benchmark problems we finish the testing of our algorithm performance with benchmark functions. We were able to solve all the proposed problem with a high probability of success. Default parameter values were used for the test, although in some cases special settings had to be considered to increase the success rate.

Problem Number	$f(x^*)$	Results with *SSm*				
		Best	**Mean**	**Worst**	**% Success**	**Mean Evaluations**
1	-40.957	-40.957	-40.957	-40.957	100	541
2	694.9	694.9	694.9	694.9	100	1043
3	37.219	37.219	37.219	37.219	100	1467
4	7.6672	7.6672	7.6672	7.6672	100	53019
5	4.5796	4.5796	4.5796	4.5796	100	843
6	0.254487	0.254484	0.254487	0.254487	100	57954
7	6.00976	6.00972	6.00975	6.00976	100	1345
8	73.0353	73.0351	73.0353	73.0353	100	4188
9	68.0097	68.0097	68.0097	68.0097	100	14925
10	-8.06414	-8.06414	-8.06414	-8.06414	100	12359
11[a]	5.3	5.3	5.3	5.3	100	30639
12	285507	285507	285507	285507	100	46526
13	0.846246	0.846246	0.940063	1.82256	53	575033
13[b]	0.846246	0.84624	0.847983	0.859276	87	202028

[a]Results in 20 runs

[b]Increasing the local search frequency

Table 4.6: Mixed-integer test problems results

Chapter 5

Improved scatter search for computationally expensive process models

One characteristic of many mathematical models describing industrial processes is that they are computationally expensive to evaluate (i.e., each simulation can take minutes, or even hours of CPU computation time on an ordinary PC). Thus, there is a need of fast algorithms in terms of a reduced number of function evaluations (i.e., simulations) to avoid unaffordable computation times for an optimization. One may consider global optimization methods which employ surrogate-based approaches to reduce computation times, and which require no knowledge of the underlying problem structure (see Section 1.2.4).

Although quadratic interpolation methods have been widely used as surrogate models in global optimization applications (Simpson et al., 2001), the recent taxonomy of surrogate-based optimization methods by Jones (2001b) indicates that the most promising techniques are those based on kriging and radial basis functions (Gutmann, 2001; Björkman and Holmström, 2000; Regis and Shoemaker, 2007a,b; Holmström, 2007). Some recent contributions compare these and other metamodeling techniques in engineering problems (Simpson et al., 2001; Jin et al., 2001; Wan et al., 2005; Egea et al., 2007c)

In this work, we will focus on kriging as a surrogate modeling technique to reduce the number of simulations in global optimization.

5.1 Kriging

5.1.1 Theory

The term kriging originates from geostatistics and the method was named and formalized by a French mathematician (Matheron, 1963) based on the Master's thesis of Daniel Gerhardus Krige (Krige, 1951). Kriging can be defined as a probabilistic interpolation method to create cheap-to-evaluate surrogate models from scattered observations minimizing the expected squared prediction error subject to being unbiased and being linear in the observations (Jones, 2001b). Many examples of kriging implementations that illustrate its superiority over other interpolation methods can be found in the literature (see for example Cox and John 1997; Jones et al. 1998; Sasena et al. 2002 and Martin and Simpson 2005).

Consider a real function, f, to be interpolated. Assume that f is a sample path of a second-order Gaussian random process denoted by F. Thus for all x, $f(x)$ is a realization of the Gaussian random variable $F(x)$. The covariance function of F plays a fundamental role since it indicates how two values of f, say $f(x)$ and $f(y)$, should be close depending on the distance between x and y. Kriging computes the best linear unbiased predictor of $F(x)$ using the observations of F on a set of points $\mathbb{S} = \{x_1, \ldots, x_n\}$. Denote by $F_{\mathbb{S}}$ the vector of observations $[F(x_1), \ldots, F(x_n)]^T$. The Kriging predictor is a linear combination of the observations, which may be written as

$$\hat{F}(x) = \lambda(x)^T F_{\mathbb{S}} \tag{5.1}$$

with $\lambda(x)$ being a vector of coefficients $\lambda_1, \ldots, \lambda_n$. These coefficients are chosen to obtain the smallest variance of prediction error among all unbiased predictors. This leads to a constrained minimization problem, which can be solved by a Lagrangian formulation (Matheron, 1963). The vector $\lambda(x)$ can be computed as the solution of the system of linear equations

$$\begin{pmatrix} K & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} \lambda(x) \\ \mu(x) \end{pmatrix} = \begin{pmatrix} k(x) \\ a(x) \end{pmatrix} \tag{5.2}$$

where K is the covariance matrix of the random vector $F_{\mathbb{S}}$, A is a matrix of known functions $a_1, \ldots, a_q$ (usually polynomials of low degree) evaluated at the points of $\mathbb{S}$, $k(x)$ is the covariance vector between $F(x)$ and $\mathbb{S}$, $a(x)$ is the vector of $a_1, \ldots, a_q$ evaluated at x, $\mu(x)$ is a vector of Lagrangian multipliers and 0 is a matrix of zeros. The computational burden of computing kriging coefficients is $\mathrm{O}(n^3 N)$, with N being the number of points in

which the kriging prediction is performed (Villemonteix et al., 2008). Thus, the computation time needed for computing the prediction in a new set of points increases in a cubic way with the number of observations.

Knowing the kriging coefficients, the predicted value of f given $f_{\mathbb{S}} = (f(x_1), \ldots, f(x_n))^T$ can be written as

$$\hat{f}(x) = \lambda(x)^T f_{\mathbb{S}} \tag{5.3}$$

5.1.2 Covariance choice

The selection of a suitable covariance function is crucial for the success and accuracy of the kriging prediction. For this purpose, it is usual to choose a parameterized covariance model and to estimate its parameters based on the observations.

The use of a stationary, isotropic covariance model with one parameter to adjust regularity makes it possible to model a large class of functions (Vazquez, 2005). Here we use the Matérn covariance, with the following parameterization (Yaglom, 1987; Stein, 1999).

$$k(h) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)} = \left(\frac{2\nu^{1/2}h}{\rho}\right)^{\nu} \mathcal{K}_\nu \left(\frac{2\nu^{1/2}h}{\rho}\right) \tag{5.4}$$

where h is the Euclidean distance between two points, $\mathcal{K}_\nu$ is the modified Bessel function of the second kind, ν controls the regularity, σ^2 is the variance and ρ represents the range of the covariance.

Before performing the kriging prediction, the parameters of the covariance must be estimated based on the observations. Assuming that the mean of $F(x)$ is zero for the sake of simplicity, this parameter estimation can be done calculating the maximum-likelihood estimate of the vector of covariance parameters, ϕ, by minimizing the negative log-likelihood (Vecchia, 1998) that can be expressed as:

$$l(\phi) = \frac{n_0}{2}\log 2\pi + \frac{1}{2}\log\det K(\phi) + \frac{1}{2} f_{\mathbb{S}}^T K(\phi)^{-1} f_{\mathbb{S}} \tag{5.5}$$

with n_0 being the number of observations, $f_{\mathbb{S}}$ the observations and $K(\phi)$ the covariance evaluated with the set of parameters ϕ. In the general case where the mean of $F(x)$ is not zero, the covariance parameters can be estimated using other methods, as for example using Restricted Maximum Likelihood (Dietrich and Osborne, 1991; Stein, 1999).

5.1.3 Illustrative examples

One of the advantages of kriging is that the variance of the prediction error at x can be computed even without any evaluation of f. This is one of the strongest points of this method compared to others: kriging provides a statistical framework that gives an idea of the uncertainty associated to each prediction. This also helps us to know which points are worth evaluating in different applications of the method (for example, in global optimization).

Figure 5.1 shows the kriging prediction of the sine function in the interval $[-10\ 10]$. The blue line is the real function whereas the black line is the kriging prediction based on the observations (red circles). For a point, x_i, kriging provides a normal distribution function (green line). The mean of the distribution is the kriging prediction and the variance is also provided in the calculation process. With this distribution we can not only know which is the prediction in every point provided some observations but also the uncertainty associated to this prediction and thus the probability of finding a value lower than a threshold when evaluating the real function.

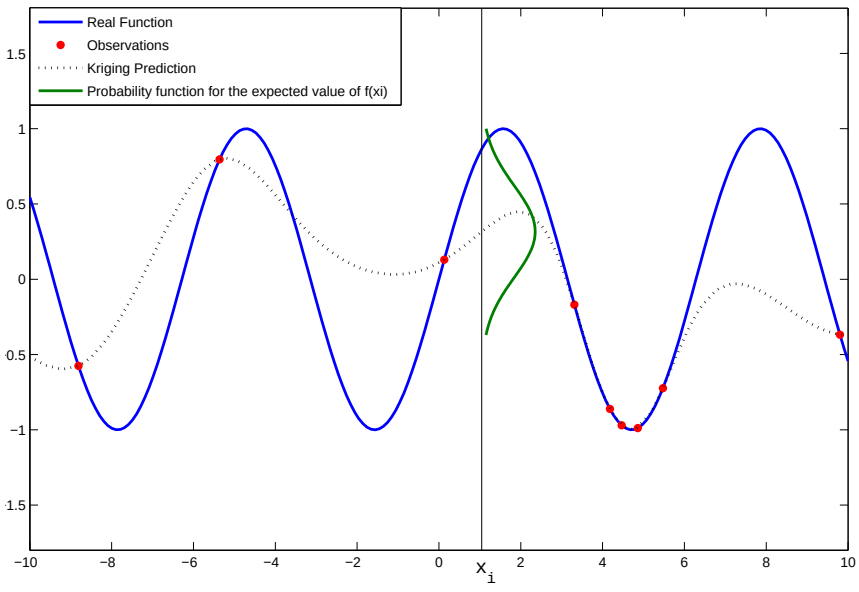


Figure 5.1: Kriging prediction and Gaussian distribution for point x_i (sine function)

A similar way of processing the statistical information provided by kriging is shown in Figure 5.2. In this case 95% confidence intervals (in green) are plotted. They give us an idea about the uncertainty of the prediction in each point of the search space. The intervals are

obviously empty in the observations and they increase as they get further from them. These intervals as well as the kriging prediction will be updated as the number of observations increases to become more and more accurate.

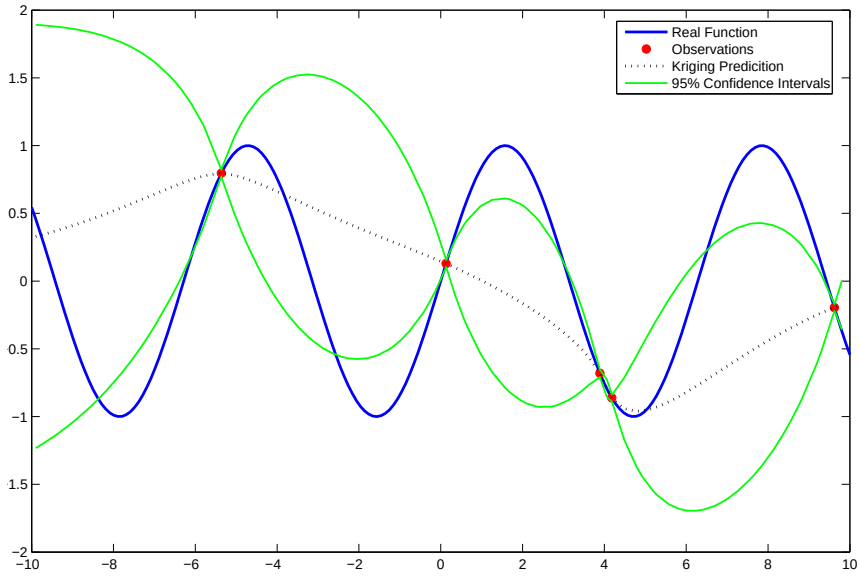


Figure 5.2: Kriging prediction and 95% confidence intervals (sine function)

5.2 *SSKm*

A new algorithm for global optimization of costly nonlinear continuous problems is presented in this section. This methodology, and its associated software tool, *SSKm* (Scatter Search with Kriging for Matlab), is able to manage this class of problems by linking a scatter search method with a kriging interpolation. Thanks to the statistical information provided by kriging, the algorithm is able to discard the evaluation of solutions that are not likely to provide high quality function values. This makes the algorithm suitable for the optimization of computationally costly problems, as it will be illustrated in Section 5.3 and in some of the optimization problems presented in the Applications part of this work.

There are a number of reasons that justify the combination of these two techniques. On the one hand, both of them have proved to be efficient in their respective fields (i.e., global optimization in the case of scatter search, and prediction in the case of kriging). On the other hand, kriging needs a careful selection of the points in which the prediction will be done (Pardalos et al., 2000), in order to avoid investing computational effort in calculating

the kriging coefficients for points that will not be of interest. Scatter search operates on a set of solution vectors that evolve during the search process (new solutions replace old ones in the *RefSet*). These solutions have, by construction, good objective function values. Thus, they are good candidates for the kriging prediction.

Since kriging provides statistical information about the prediction, it can be used to create a performance index that may help us decide which points should be evaluated. In some sense, this information is acting as a filter, thus the scatter search algorithm may not contain all the advanced strategies presented in Chapter 4. We present some alternatives specially designed to interact with the kriging module:

- The *log_var* strategy (see Section 4.2.1) has been implemented. This log-normalization may also help the prediction phase with kriging allowing a smoother correlation among variables.

- Combinations are based on hyper-rectangles around the solutions to be combined (Section 4.2.3).

- The *Refset Update Method* takes into account an Euclidean distance to avoid duplication and clustering inside it (Section 4.2.4). This Euclidean distance allows the method to replace other solutions in *RefSet* different from the worst one as long as the distance is not violated. In this case the Euclidean distance is static since the conditions of the dynamic distance defined in Chapter 4 (e.g., many function evaluations per iteration) do not apply here.

- The *RefSet* rebuilding strategy based on orthogonality (Section 4.2.6) is used together with the classical one based on distances.

Other specific characteristics of *SSKm* are the following:

- No *Improvement Method* has been implemented since the purpose of the algorithm is to reduce the number of function evaluations as much as possible to locate the global optimum.

- One of the most important aspects of using kriging for global optimization is the selection of the points in which the prediction will be performed. A straightforward strategy is to select a uniform set of points within the search space, but this would give every

area the same importance. Besides, we would need a huge number of points not to miss any promising area, thus increasing the computation time too much. A selection of points guided by an evolutionary algorithm such as scatter search ensures a good balance between intensification and diversification to avoid spending too much time in poor areas but ensuring some diversity. In this case the *RefSet* dimension (and thus the number of solutions generated) is higher than in classical scatter search implementations since we want to cover a big area of the search space for the kriging prediction. In our algorithm, we use a constant number of elements in the *RefSet* without using memory (i.e., a constant number of generated solutions for the prediction in each iteration).

- Since there is only one evaluation per iteration, the *RefSet Update Method* is different than the one described in Chapter 4. Instead of selecting the best b elements which comply with the filters among parents and children to generate the new *RefSet*, the replacement is done on a one-by-one basis. When a new solution is evaluated, its Euclidean distances to all the *RefSet* members are computed. If all these distances are bigger than a fixed threshold, the new solution replaces the worst solution in *RefSet* in terms of quality. If only one these distances violates the threshold but the new evaluated solution is better in terms of quality than the *RefSet* solution close to it, the former replaces the latter. Otherwise, there is no replacement in the current iteration.

- Many computationally expensive models involve numerical integration of set of ODE's or DAE's. For different reasons, these integrations can be numerically unstable using some sets of decision variables values. This can cause that the simulation does not provide any value for the objective function. In traditional optimization methods, this can be overcome by assigning a high value to the vector producing the simulation error. Thus, it is automatically discarded for the next iteration (in minimization problems). In surrogate model-based optimization this strategy is not adequate since the prediction method makes use of all the evaluated vectors and their corresponding function values. A wrong value of the function value may result in a very inaccurate surface not only in that part of the search space but also in others. To avoid this, our algorithm discards the vectors which produce simulation errors, not adding them to the observations.

- To keep track of the evaluations skipped due to the kriging information, the algorithm saves the number of function evaluations that a classical scatter search implementation

using the same settings as our method would have done in the same problem.

- When the sign of the objective function value does not change for a problem and there might be different orders of magnitude among objective function values, the user is recommended to use a log-normalization of the function value (named *log_f* in the options) to help the prediction be smoother.

The termination criteria are the same as in *SSm*: the algorithm can stop either by number of function evaluations, by computation time or by reaching a specified function value. In Appendix A, the help file of the Matlab implementation of our algorithm is shown. Algorithm 17 shows the *SSKm* procedure in pseudocode.

Algorithm 17 *SSKm* algorithm

Generate diverse solutions and evaluate them as the first observations
Form the first *RefSet*
Compute the best observation, f_{best}
Set $estimation = 1$
while not termination criterion **do**
  Generate solutions by combinations of pairs of solutions in the *RefSet*
  if *estimation* **then**
    Estimate covariance parameters
  end if
  Compute kriging prediction and variance over generated solutions
  Compute the performance index for each solution
  Evaluate the solution, x_{new} with the maximum value of the performance index
  if diff(*estimation*, real value) $< \epsilon$ **then**
    $estimation = 0$
  else
    $estimation = 1$
  end if
  Add $(x_{new}, f(x_{new}))$ to the observations
  Update f_{best}
  Check Euclidean distances of the evaluated solution to the *RefSet* members
  Sort Euclidean distances and *RefSet* by increasing distance values
  if $distances_1 \geq dth$ **then**
    Replace the worst solution in *RefSet* by the new observation, x_{new}
  else if $distances_i \geq dth$ with $i \in [2, 3, \ldots, b]$ **and** $f(x_{new}) < f(x^1_{RefSet})$ **then**
    Replace x^i_{RefSet} (or the worst among them if there is more than one) by x_{new}
  end if
  Check termination criterion
end while

5.2.1 Selection of a performance index for evaluating new points

One of the key points of the application of surrogate models in global optimization is to choose the next point to be evaluated. This is a very important task since the budget in the

number of simulations when dealing with computationally expensive models is usually low. Thus, the criterion for selecting the new points to evaluate must be robust and efficient.

Kriging provides fundamental information to create a performance index which helps us choose new points to be evaluated. In a simple approach, we could select the point whose kriging prediction is the best in terms of quality. However, in the first stages of the procedure, when the number of observations is low, the kriging prediction might not be accurate enough (see Figure 5.2), leading to important under- or over-estimations of the real function values. The second parameter provided by the kriging is the variance of the prediction in every point (i.e., the uncertainty associated to the prediction in every point). Choosing the variance as a performance index does not seem to be a suitable option since it only gives information about the relative distance of a point from the observations (and this can be qualitatively substituted just by calculating Euclidean distances, thus saving the computational cost of the kriging prediction).

Hence, an adequate performance index for the selection seems to be a combination between these two parameters. The most relevant contributions about kriging for global optimization use both parameters for defining their criteria to choose the new evaluated points. The most popular criterion is the *EGO* algorithm (Jones et al., 1998) based on the *expected improvement, EI*, which computes how much improvement is expected with respect to the best observation so far if the function is evaluated in a new point. The *EI* is defined as:

$$EI(x) = \hat{\sigma}(x)[u\Phi(u) + \phi(u)] \tag{5.6}$$

with

$$u = \frac{f_{min} - \hat{f}(x)}{\hat{\sigma}(x)} \tag{5.7}$$

where $\hat{f}(x)$ is the kriging prediction in the point x and $\hat{\sigma}(x)$ the variance of this prediction. f_{min} is the best observation so far (in a minimization problem). $\Phi()$ and $\phi()$ denote the cumulative distribution function (cdf) and the probability density function (pdf) of the standard normal distribution, respectively.

The *EI* is the most used criterion for evaluating new points when using kriging as a prediction method. Several modifications of this formulation have been proposed since its foundation, like for example the one by Huang et al. (2006) to deal with stochastic models or the *generalized expected improvement* (Sasena et al., 2002), formulated as:

$$EI^g(x) = \hat{\sigma}(x) \sum_{k=0}^{g} (-1)^k \left(\frac{g!}{k!(g-k)!} \right) u^{g-k} T_k \tag{5.8}$$

where

$$T_k = -\phi(u) u^{k-1} + (k-1) T_{k-2} \tag{5.9}$$

starting with $T_0 = \Phi(u)$ and $T_1 = -\phi(u)$. u is the same expression as in Equation 5.7

The parameter g controls the intensification and diversification of the search. There is no obvious method to select a proper value for g. Sasena et al. (2002) proposed a heuristic method called *cooling schedule* to change the value of g during the search depending on the number of iterations, making the search more global in the first stages and more local at the end.

Other authors have used similar criteria. Egea et al. (2007b) used the probability of improving the best observation to date as a selection criterion in their scatter search-based algorithm. This is equivalent to the *generalized expected improvement* with $g = 0$. The point to be evaluated is the one maximizing $\Phi(u)$. Davis and Ierapetritou (2007) balanced between intensification and diversification by sampling points in three different sets: points with high variance, points with good prediction values and points in which the kriging predictions between two consecutive iterations are very different. Since they sample the same number of points in every set in each iteration, this involves a minimum of 3 new evaluated points by iteration. Villemonteix et al. (2008) presented a novel criterion based on a rigorous statistical framework in which the following point to be evaluated is the one minimizing the uncertainty on the location of the global optimum. This parameter-free strategy automatically balances between intensification and diversification making use of the *stepwise uncertainty reduction strategy*.

In this work, we propose another heuristic method which combines high variance with probability of improving the best value so far to achieve a suitable balance between intensification and diversification. Like in the *cooling schedule* of Sasena et al. (2002), the search is more focused on diversification in the first iterations and more focused in intensification in the last iterations. Unlike in the *cooling schedule*, in which the value of g decreases depending on the number of iterations in a discrete way, here the variation is continuous. We define the following performance index for each point in which the kriging prediction is done, in order to choose the one maximizing it to perform the next function evaluation.

$$PI_i = w\Phi(u_i) + (1-w)\frac{\hat{\sigma}_i}{\hat{\sigma}^{max}} \tag{5.10}$$

where $i \in [1, 2, \ldots, N]$ with N being the number of points in which the kriging prediction is performed. $\hat{\sigma}^{max}$ is the highest predicted variance among all these points and divides the predicted variance of every point in N in order to normalize the second adding term of Equation 5.10 and give it the same importance as the first term, which is a probability and therefore it varies between 0 and 1. The weight w is controlling which term has the best importance along the optimization procedure. A low value of w gives more importance to the variance term (i.e., to diversification) whereas a high value of it focuses on maximizing the probability of improving the best solution found (i.e., on intensification). In a general case, w has the form of an increasing continuous function which depends on the progress of the search. We propose a general exponential form for w as follows:

$$w = \left(\frac{n}{n_f}\right)^p \quad \text{or} \quad w = \left(\frac{t}{t_f}\right)^p \tag{5.11}$$

where n is the number of function evaluations done so far and n_f is the maximum number of function evaluations allowed. Similarly, t is the computation time consumed so far and t_f is the maximum computation time allowed. w will have one or other form depending on the stopping criterion selected. p is a positive real number which determines the balance between intensification and diversification. Values of p close to 0 give more importance to the intensification term from the beginning of the search and they are recommended for convex problems or for problems in which the budget of function evaluations (or computation time) is small. High values of p focus the search onto points with high predicted variance focusing on intensification only in the last iterations. These high values are recommended for highly multimodal problems or for applications with a large budget of simulations (or computation time). In this work, we have used values of p between 0 (aggressive strategy) and 1 (robust strategy). We have experimentally found that an intermediate value such as $p = 0.5$ provided good balances during the search.

Figures 5.3 and 5.4 show how the next point to be evaluated is selected depending on the stage of the search and the value of p. Figure 5.3 shows the optimization of a unidimensional multimodal function with a budget of 20 function evaluations with an initial (uniform) sampling of 5 observations. The blue line represents the real function. Observations are the red solid circles and the kriging prediction is denoted by the black dotted line. In the sub

plot below, the performance index for the points in which the prediction has been done are shown for different values of p. For a value of $p = 0$ (black line, aggressive search) the point which maximizes the performance index is very close to the best observation (it focuses on intensification). For a higher value of p ($p = 1$, red line) the point with the maximum performance index (i.e., the lower bound of the decision variable) is different than in the previous case. For the sake of comparison, we calculated the *expected improvement* of the same points in which the kriging prediction has been done. The figure shows how its maximum value is located in between the two best observations, thus not searching as globally as in the case of our algorithm when $p = 1$.

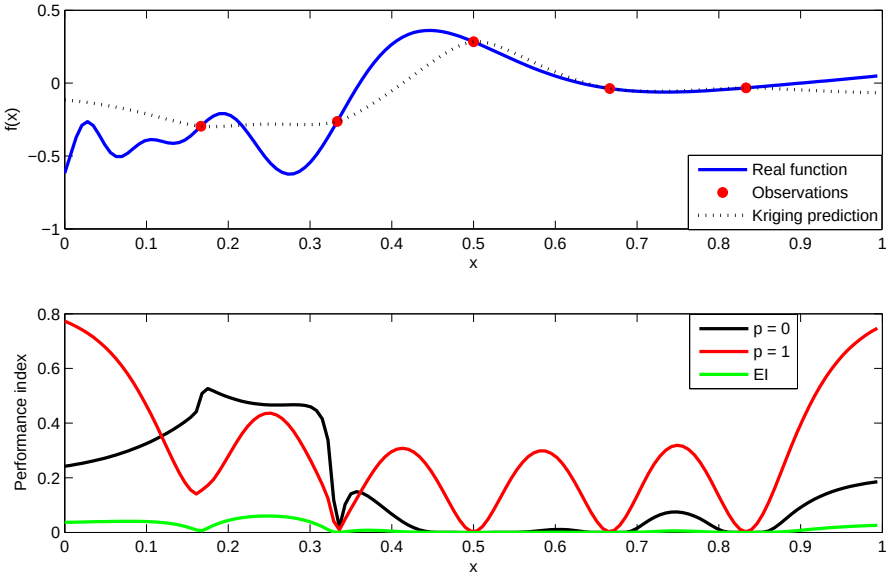


Figure 5.3: Performance index calculation for $n/n_f = 0.25$

Figure 5.4 shows the same example but having 15 initial uniformly distributed observations (for a total budget of 20 function evaluations). In this case, the value of n/n_f is closer to 1 and the performance indexes calculated with different values of p tend to be maximal in the same point. Indeed, it can be checked that at this stage of the search the performance indexes corresponding to $p = 0$ and $p = 1$, shown in the figure, have their global maximum in the same point of the search space. For the last iterations of the search, the values of p have less and less importance and the value of the weight w converges to 1. In this case, the *expected improvement* is maximized in the same area.

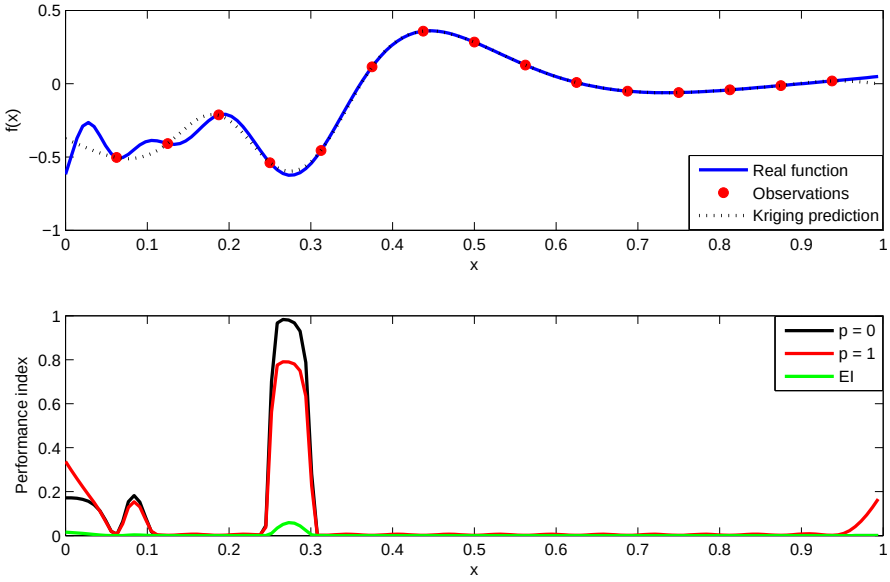


Figure 5.4: Performance index calculation for $n/n_f = 0.75$

5.3 Application examples

5.3.1 Kriging prediction

In this section we will illustrate a kriging interpolation using a different number of initial observations in the *Michalewicz* function (Michalewicz, 1992), defined by:

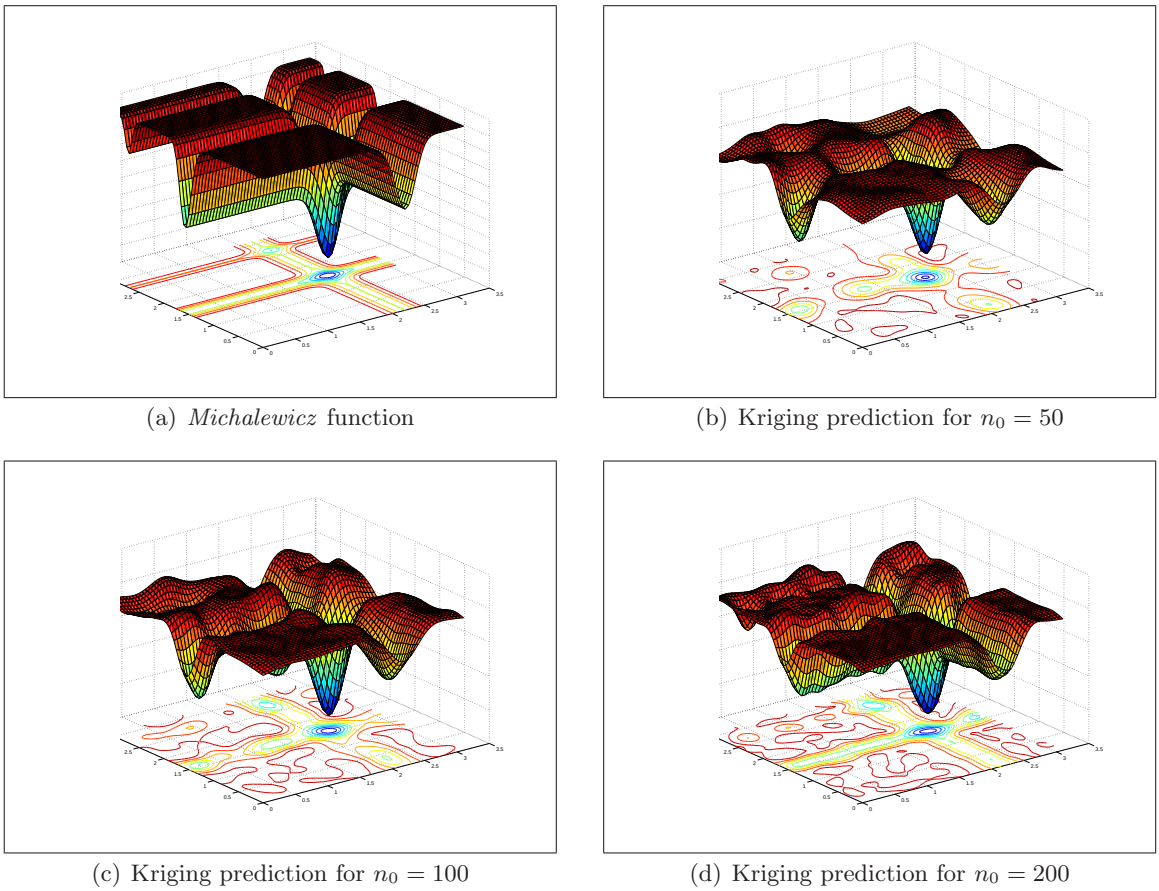
$$F(\mathbf{x}) = -\sum_{i=1}^{2} \sin(x_i)(\sin(ix_i^2/\pi))^{20} \tag{5.12}$$

in the interval $[0, \pi]$ for both x_1 and x_2.

Both the real function and kriging predictions for a different number of observations are presented in Figure 5.5. The real function is plotted in Figure 5.5(a). Figures 5.5(b), 5.5(c) and 5.5(d) plot the kriging prediction of the function in the same interval using $n_0 = 50$, 100 and 200 observations (i.e., real function evaluations) within the same interval, respectively. It can be observed that the larger the number of observations, the higher accuracy in the prediction.

5.3.2 Kriging-based global optimization

In this section, illustrative examples of the application of our algorithm for global optimization are shown. We consider two benchmark problems to test the efficiency of our *SSKm* solution



(a) *Michalewicz* function

(b) Kriging prediction for $n_0 = 50$

(c) Kriging prediction for $n_0 = 100$

(d) Kriging prediction for $n_0 = 200$

Figure 5.5: *Michalewicz* function and its kriging approximation using a different number of observations

method. For the sake of comparison, we apply two traditional optimization algorithms, one deterministic, *DIRECT* (Jones, 2001a), and one stochastic, *Differential Evolution* (*DE*, Storn and Price 1997), and two algorithms making use of surrogate models: *rbfSolve* (using radial basis functions interpolation) and *ego* (using kriging interpolation), both included in the Tomlab optimization toolbox (Holmström and Edvall, 2004). To test the new features of our algorithm with respect to a previous advanced scatter search implementation we have also applied the algorithm *SSm*.

The *six-hump camel-back* function (i.e., function 9 in Appendix B) has several local minima and two global optima. For solving this problem we have applied our algorithm to this function in the search space defined by the bounds [-1.9, -1.1] and [1.9, 1.1] by using an initial sampling of 20 diverse points and calculating 30 extra points for a total number of 50 points. A total number of 50 function evaluations were also fixed for the other algorithms and the same initial 20 points were used in all cases.

Table 5.1 shows the values of the two global optima as well as the closest points to them achieved by each algorithm. Figure 5.6 presents the contour plots of the function and the 30 last points evaluated by each algorithm, represented as triangles (except for *ego*, which stopped after 20 extra function evaluations).

Solver	**GO_1**		**GO_2**	
	x^*	$f(x^*)$	x^*	$f(x^*)$
	[-0.0898, 0.7126]	**-1.0316**	**[0.0898, -0.7126]**	**-1.0316**
DIRECT	[-0.1407, 0.7333]	-1.0191	[0.1407, -0.7333]	-1.0191
DE	[0.1900, 0.6233]	-0.6902	[0.2956, -0.7333]	-0.8774
SSm	[0, 0.7196]	-0.9987	[0, -0.6794]	-0.9941
SSKm	[-0.0891, 0.7089]	-1.0315	[0.0706, -0.6973]	-1.0314
rbfSolve (Thin plate splines)	[-0.1068, 0.6678]	-1.0143	[0.0846, -0.7034]	-1.0309
rbfSolve (Cubic splines)	[-0.1251, 0.6544]	-0.9992	[0.0882, -0.7105]	-1.0316
ego	[-0.0853, 0.7142]	-1.0315	[0.0851, -0.7121]	-1.0315

Table 5.1: Solutions for the optimization of the *six-hump camel-back* function in 50 function evaluations

Table 5.1 and Figure 5.6 demonstrate the superiority of surrogate model-based algorithms over the other three algorithms because they do not only achieve the best function values but also locate most of the evaluations in the neighborhood of both global minima, whereas the other methods present a bigger dispersion on their evaluations. *SSKm* presents the lowest dispersion within its solutions whereas *rbfSolve* and *ego* evaluate several points touching the bounds of the decision variables. This also shows the ability of surrogate model-based optimization algorithms, not only for locating one global minimum, but also for locating several global minima when they exist. This is a very interesting characteristic for robust optimization purposes.

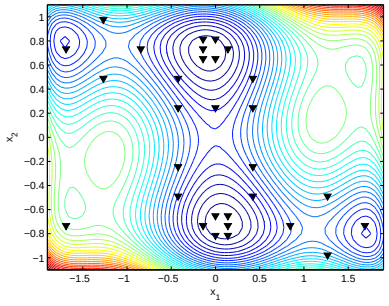
For the *six-hump camel-back* example, a value of $p = 0$ in *SSKm* was enough for finding both global minima given that initial sampling. In order to illustrate how the algorithm performs depending on the value of p a new experiment was carried out over another multimodal function: the *Branin* function (i.e., function 1 in Appendix B). This function has three global optima and, depending on the value of p, *SSKm* should be able to locate one or all of them. The procedure was analogous as in the previous example: a set of 20 function evaluations (red circles in the figures) were used as initial observations and the algorithms performed 30 new evaluations (black triangles), shown in Figure 5.7. In this case, the comparison was carried out using only the surrogate model-based algorithms (i.e., *rbfSolve* and *ego*).

A value of $p = 0$ makes *SSKm* find two of the global minima. As we increase the value of p, the new evaluations are being more scattered. For $p = 0.5$ the algorithm is able to

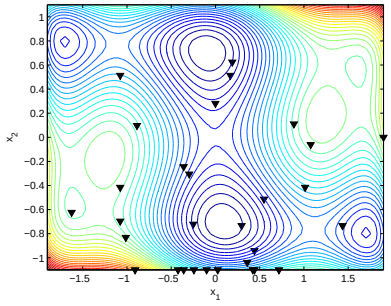
locate the three global minima. In the case of $p = 0.75$, the results are very similar but more "outliers" start to appear near the bounds, which means that the algorithm is searching more globally. The other surrogate model-based algorithms, *rbfSolve* and *ego* select more scattered points to evaluate by default (i.e., similar to *SSKm* with $p = 0.75$), which results in a robust search. Both *rbfSolve* (using cubic splines) and *ego* locate the three global optima in the budget of function evaluations fixed.

5.4 Conclusions

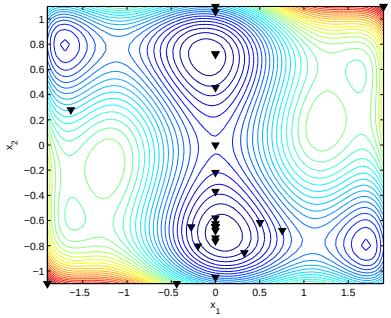
In this chapter we have presented an improved scatter search algorithm combining a scatter search procedure with a kriging interpolation. It has been specially designed for the optimization of computationally expensive process models, in which the budget of simulations is usually low. The scatter search phase selects a set of high quality solutions which evolve whereas the kriging interpolation provides statistical information about the quality and uncertainty associated with each point, in order to choose the next observation (i.e., function evaluation). We have proposed a performance index using the information provided by kriging which determines the next point to evaluate. The method increases the intensification in the last iterations to refine the best solution found, but it can be tuned to perform a more diverse search at the beginning (more suitable for multimodal problems) or to focus directly on the intensification (recommended for unimodal problems). The method has been tested and compared with other global optimization algorithms over two multimodal benchmark functions, showing its ability to locate all the global optima in a small number of function evaluations. As expected, comparisons show that surrogate model-based optimization algorithms need less function evaluations to locate the global optimum of a function than other global optimization methods.



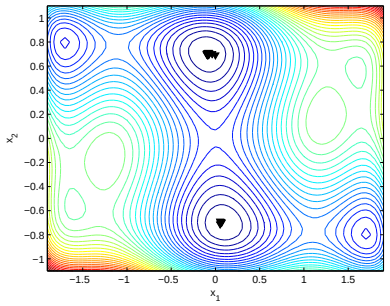
(a) *DIRECT*



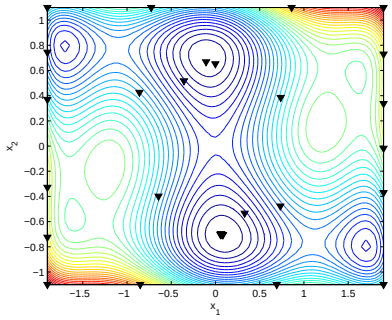
(b) *DE*



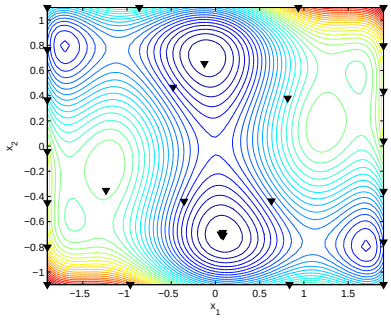
(c) *SSm*



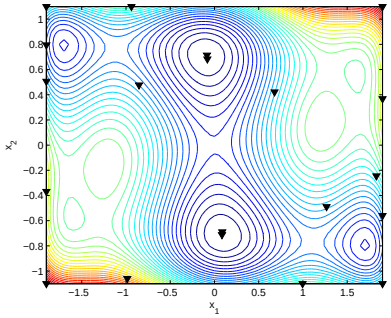
(d) *SSKm*



(e) *rbfSolve* (Thin plate splines)

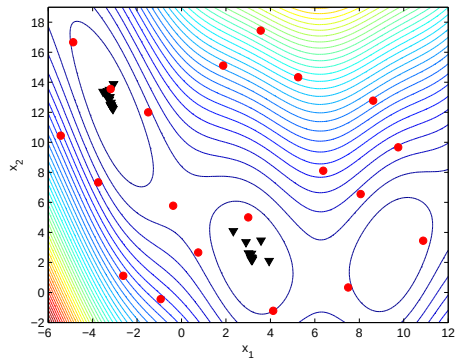


(f) *rbfSolve* (Cubic Splines)

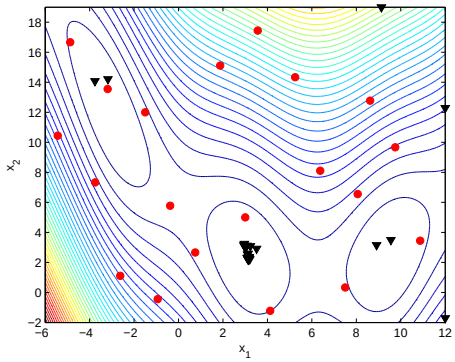


(g) *ego*

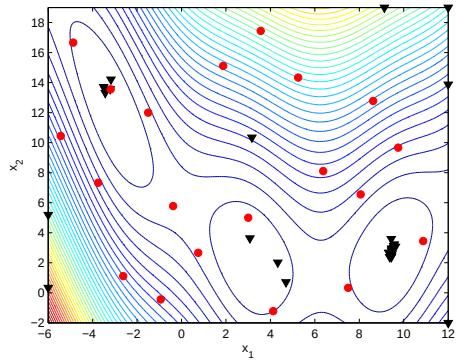
Figure 5.6: Contour plot of the *six-hump camel-back* function with the evaluations done by different global optimization algorithms



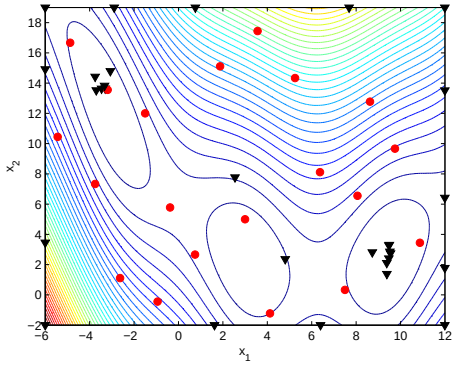
(a) *SSKm* ($p = 0$)



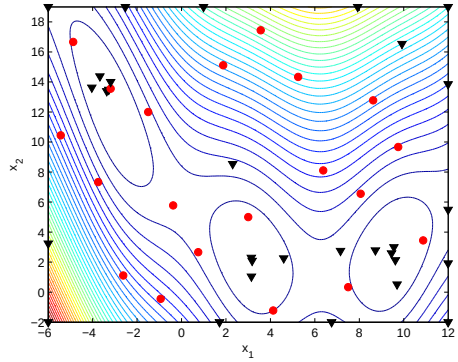
(b) *SSKm* ($p = 0.5$)



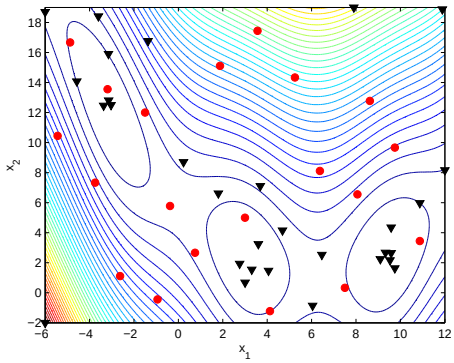
(c) *SSKm* ($p = 0.75$)



(d) *rbfSolve* (Thin plate splines)



(e) *rbfSolve* (Cubic splines)



(f) *ego*

Figure 5.7: Contour plot of the *Branin* function using 20 initial observations (red circles) and 30 extra evaluations (black triangles)

Part III

Applications

Chapter 6

Preliminary chapter

In the following chapters, a set of bioprocess optimization problems covering the different types presented in Chapter 1 (i.e., parameter estimation, integrated design and control, and dynamic optimization) will be used as case studies to test the performance of the algorithms proposed in this work. This preliminary chapter presents the optimization methods used for comparison with our algorithms and the procedure used to carry out the experimental analysis.

6.1 Selected optimization methods

A set of different state-of-the-art global optimization methods has been selected to compare their results with those obtained with the algorithms proposed in this study.

- **CMAES:** *Covariance Matrix Adaptation Evolutionary Strategy.* This is an evolutionary algorithm that makes use of the covariance matrix in a similar way to the inverse Hessian matrix in a quasi-Newton method. This can improve the performance on ill-conditioned problems by orders of magnitude (Hansen et al., 2003).

- **DE:** *Differential Evolution.* This is a heuristic algorithm for the global optimization of nonlinear and (possibly) non-differentiable continuous functions presented by Storn and Price (1997). This population-based method handles stochastic variables by means of a direct search method which outperforms other popular global optimization algorithms, and it is widely used by the evolutionary computation community.

- **SRES:** *Stochastic Ranking Evolutionary Strategy.* This algorithm consists of a (μ, λ) evolutionary strategy combined with an approach to balance objective and penalty functions stochastically (Runarsson and Yao, 2000). In the (μ, λ)-ES algorithm, the evaluated objective and the penalty functions for each individual are used to rank the individuals in a population, and the best (highest ranked) μ individuals out of λ are selected for the next generation. This feature makes it especially appealing for the case of constrained problems.

- **DIRECT:** *DIviding RECTangles.* This is a deterministic global optimization algorithm based on a modification of the Lipschitzian optimization scheme to solve difficult global optimization problems (Jones, 2001a). The search is performed by dividing the space into hyper-rectangles and is specifically designed for those cases in which the objective function is non-smooth, no derivative information is available, or its evaluation requires several different simulations to be performed. The algorithm operates by systematically dividing the optimization domain into hyper-rectangles, and evaluating the objective function in their centers. There are two phases to an iteration of *DIRECT*: first, hyper-rectangles are identified as potentially optimal (i.e., it is expected that they contain a global solution); the second phase consists of dividing potentially optimal hyper-rectangles into smaller ones. The objective function is evaluated in the centers of new hyper-rectangles and the search is directed towards unexplored regions of the domain. We will use *glcDirect*, the *DIRECT* implementation of Holmström and Edvall (2004) in our computational testing.

- **OQNLP:** *OptQuest-NLP.* This is and optimization engine released by OptTek Systems, Inc. As described in Laguna and Martí (2002), this is a generic optimizer that overcomes the deficiency of black box systems and successfully embodies the principle of separating the method from the model. In such a context, the optimization problem is defined outside the complex system. Therefore, the evaluator can change and evolve to incorporate additional elements of the complex system, while the optimization routines remain the same. Hence, there is a complete separation between the model used to represent the system and the procedure that solves optimization problems formulated around the model. The optimization technology embedded in this algorithm is also scatter search. The method is organized to (1) capture information not contained sepa-

rately in the original points, (2) take advantage of auxiliary heuristic solution methods (to evaluate the combinations produced and to actively generate new points), and (3) make dedicated use of strategy instead of randomization to carry out component steps. In our testing we use the implementation known as *OQNLP* (Ugray et al., 2005) which uses OptQuest to provide starting points for any gradient-based local NLP solver. This procedure combines the superior accuracy and feasibility-seeking behavior of gradient-based local NLP solvers with the global optimization abilities of scatter search. In a recent review by Neumaier et al. (2005) comparing several GO solvers over a set of 1000 constrained GO problems, *OQNLP* obtained the best performance among the stochastic methods. Furthermore, it solved the highest percentage of problems with a high number of decision variables.

- **rbfSolve:** This algorithm, included in the Tomlab optimization toolbox (Holmström and Edvall, 2004), solves costly global optimization problems using a radial basis functions interpolation algorithm (Gutmann, 2001; Björkman and Holmström, 2000). It fits a response surface (based on splines) from data collected by evaluating the objective function at some points and then applies a global optimization algorithm, *glcFast* (Holmström and Edvall, 2004), which is a *DIRECT* implementation, and a local algorithm, *npsol* (Gill et al., 1998), using different initial points over that surrogate model. The first set of points to create the response surface may be given by the user or selected by the algorithm based on different strategies.

- **ego:** Also included in the Tomlab optimization toolbox, this algorithm solves costly global optimization problems using the Efficient Global Optimization (*EGO*) algorithm (Jones et al., 1998), based on kriging interpolation. The idea of the *EGO* algorithm is to first fit a response surface to data collected by evaluating the objective function at a few points. Then, *EGO* selects those points maximizing *the expected improvement* (see Section 5.2.1) to be evaluated.

In the following chapters, the performance of these global optimization algorithms in a set of bioprocess engineering optimization problems will be compared with the performance of the algorithms proposed in this work. In the next section, some notes regarding the procedure followed to carry out this experimental part are depicted. They explain the protocol used for doing the tests as well as other details to take into account, and they should be read before

the following chapters of this work.

6.2 Procedure followed in the experiments

In this section we highlight some aspects of the procedure followed to test the case studies presented in the following chapters. The same protocol was used in every problem. The reader should take into account the following points:

- For every problem, a multistart procedure with the SQP method *fmincon* (*n2fb* in double precision for the case of parameter estimation problems) was performed over a set of diverse initial points (usually 100) in order to check the practical multimodality of the problem. It consists in applying the local search algorithm to a set of initial points uniformly distributed within the bounds (including the initial point used for every problem).

- A fixed number of function evaluations was set for every problem. These numbers are shown in Table 6.1.

Problem	Section	Maximum evaluations
Parameter estimation		
α-pinene	7.1	10000
HIV	7.2	30000
3-step pathway	7.3	20000
Integrated design and control		
Constrained WWTP	8.2	15000
COST WWTP PI tuning	8.3.2	400
COST WWTP op. design	8.3.3	800
COST WWTP MINLP	8.3.3	1500
Dynamic optimization[a]		
Ethanol	9.2	20000, 40000, 60000
Penicillin	9.3	55000, 90000, 250000
Drying	9.4	20000, 60000, 200000
Microwave	9.5	12000, 40000, 60000

[a]Three discretization levels per problem.

Table 6.1: Maximum number of function evaluations fixed for every problem

- For stochastic solvers (i.e., *CMAES*, *DE*, *SRES* and *SSm*) a set of 10 runs was performed. The *OQNLP* implementation that we used in this study does not allow to change the seed of the random numbers generator. The best, mean (as recommended

by Birattari and Dorigo 2007) and worst objective function value found in all the optimization are reported.

- For the best objective function value found, its corresponding decision vector together with the variable bounds used in every problem are also reported.

- Convergence curves showing the convergence rate of every solver to their best found solutions are presented (usually in log-scale). In order to have a fair comparison among every solver's efficiency, the same initial point (usually the middle point between the variable bounds) was used, except for *glcDirect*, which does not accept any user's given initial point.

- Default parameters were used for our proposed algorithms, *SSm* and *SSKm* (see Appendix A). For some problems, parameter values different from defaults ones are used. In those cases, they are explicitly mentioned.

- For the rest of solvers, default parameters provided by their authors are also used. In case of the population-based algorithms *DE* and *SRES*, some parameters have to be chosen by the user. The chosen parameters are shown in Table 6.2.

DE	*SRES*
VTR = -inf	$\lambda = 10*nvar$
NP = $10*nvar$	$\mu = \overline{\lambda/7}$
F = 0.85	pf = 0.45
CR = 1	varphi = 1
strategy = 3	

Table 6.2: Selected parameters for *DE* and *SRES*

- For solvers not handling constraints (i.e., *CMAES* and *DE*) a modification of the scripts was carried out to allow them to be applicable to constrained problems. In particular, the implemented strategy was the same as the one used by *SSm*. Thus, a static penalty term is added to the objective function value: the maximum absolute violation of the constraints multiplied by 10^6 is added to the actual function value. In every case, a maximum constraint violation of 10^{-5} is considered as acceptable.

Chapter 7

Parameter estimation problems

The concept of estimating parameters may be rather confusing if we take into account that a parameter is considered as a constant in mathematical models. Therefore, estimating a constant does not seem to make sense. When we talk about estimating parameters, we consider them as the decision variables in the optimization procedure, but as constants during the process which is being simulated. A typical example is the estimation of constant rates in (bio)chemical reactions: the estimated values (among the possible set of values defined by their bounds), will be used as constants and will determine the value of other variables (i.e., species concentrations) which are not calculated in the optimization procedure (even if they can define some kind of constraint which may have an influence in the optimization).

Estimating the parameters of a nonlinear dynamic model is more difficult than for the linear case, as no general analytic result exists. Biological models are often dynamic and highly nonlinear. Thus, in order to find the estimates, we must resort to nonlinear optimization techniques where a measure of the distance between model predictions and experimental data is used as the optimality criterion to be minimized. The criterion selection will depend on the assumptions about the data disturbance and on the amount of information provided by the user.

When estimating parameters of dynamical systems a number of difficulties may arise, like convergence to local solutions if standard local methods are used, very flat objective function in the neighborhood of the solution, over-determined models, badly scaled model functions or non-differentiable terms in the systems dynamics (Schittkowski, 2002).

Due to the nonlinear and constrained nature of the systems dynamics, these problems are very often multimodal (non-convex). Thus, traditional gradient based methods, like Levenberg-Marquardt or Gauss-Newton, may fail to identify the global solution and may converge to a local minimum when a better solution exists just a small distance away. Moreover, in the presence of a bad fit, there is no way of knowing if it is due to a wrong model formulation, or if it is simply a consequence of local convergence. Thus, there is a distinct need for using global optimization methods which provide more guarantees of converging to the globally optimal solution. Advances in global optimization for parameter estimation of dynamic systems have been recently made in both deterministic (Esposito and Floudas, 2000b; Lin and Stadtherr, 2006) and stochastic (Moles et al., 2003b; Rodríguez-Fernández et al., 2006a,b) methods.

The importance of using global optimization methods for parameter estimation in biological systems has been increasingly recognized in recent years (Moles et al., 2003b; Zwolak et al., 2005; Tsai and Wang, 2005; Polisetty et al., 2006).

7.1 Isomerization of α-pinene

7.1.1 Introduction

In this case study, we want to estimate 5 rate constants, $(p_1, ..., p_5)$, of a complex biochemical reaction originally studied by Box et al. (1973), which is also part of COPS (Collection of large-scale Constrained Optimization ProblemS), maintained by Dolan et al. (2004).

Figure 7.1 contains the proposed reaction scheme for this homogeneous chemical reaction describing the thermal isomerization of α-pinene (y_1) to dipentene (y_2) and allo-ocimene (y_3) which in turn yields α- and β-pyronene (y_4) and a dimer (y_5).

This process was studied by Fuguitt and Hawkins (1947), who reported the concentrations of the reactant and the four products at eight time intervals. If the chemical reaction orders are known, then mathematical models can be derived giving the concentration of the various species as a function of time. Hunter and McGregor (1967) assumed first-order kinetics and derived the following linear equations for the five responses:

$$\frac{dy_1}{dt} = -(p_1 + p_2)y_1 \tag{7.1}$$

$$\frac{dy_2}{dt} = p_1 y_1 \tag{7.2}$$

$$\frac{dy_3}{dt} = p_2 y_1 - (p_3 + p_4) y_3 + p_5 y_5 \tag{7.3}$$

$$\frac{dy_4}{dt} = p_3 y_3 \tag{7.4}$$

$$\frac{dy_5}{dt} = p_4 y_3 - p_5 y_5 \tag{7.5}$$

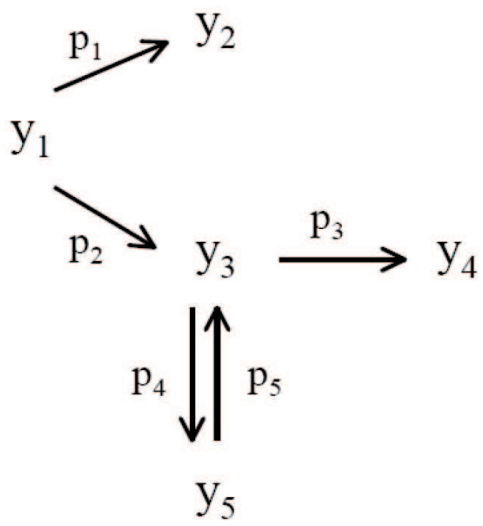


Figure 7.1: Mechanism for thermal isomerization of α-pinene

Assuming this model to be appropriate, the initial conditions for the five species are known, and we can estimate the unknown coefficients, $p_1, ..., p_5$, by minimization of a cost function which is usually a weighted distance measure between the experimental values corresponding to the measured variables and the predicted values for those variables. For this problem the cost function can be formulated as:

$$J(p) = \sum_{j=1}^{5} \sum_{i=1}^{8} (y_j(p, t_i) - \tilde{y}_{ji})^2 \tag{7.6}$$

Box et al. (1973) tried, in a first instance, to solve this problem without analyzing the multiresponse data, finding parameter values which provided an unsatisfactory data fit. Since ignoring possible dependencies among the responses can lead to difficulties when estimating the parameters (e.g., multiple local minima, very flat objective function, etc.), they described a method for detecting and handling these linear relationships. They showed that there are dependencies in the data, and, thus, only three independent linear combinations of the five responses are used in the identification, improving the fit of the data significantly. This

analysis of multiresponse data, although efficient, requires a considerable effort specially to uncover the dependencies causes once they have been found, and a deep understanding of the model (that can no longer be considered as a black-box) is essential. Moreover, it becomes unaffordable when the model complexity increases.

Tjoa and Biegler (1991) also considered this problem and used a robust local estimation approach to estimate the unknown parameters. They considered the entire data set in order to assess the performance of this method with dependencies in the data, finally reaching the same parameters reported by Box et al. (1973). However, the initial value considered for the parameters was very close to the truly optimal solution, which explains why this local method reached the global optimum without getting trapped in a local solution. As pointed out by Averick et al. (1991), the solution of this problem is not difficult to obtain from initial values of p which are close to the global solution, but it becomes increasingly difficult to solve from more remote starting points.

7.1.2 Numerical results

The lower bounds considered for the five parameters arise from physical considerations, $p_i = 0$, and we took the upper bounds to be $p_i = 1$, far from the best known solution ($p_1 = 5.93e-5$, $p_2 = 2.96e-5$, $p_3 = 2.05e-5$, $p_4 = 2.75e-4$, $p_5 = 4.00e-5$, $f(\mathbf{p}) = 19.872$). As initial point, we chose $p_i = 0.5$. The histogram resulting from the multistart procedure is shown in Figure 7.2.

The multistart procedure achieves the global solution 3 times out of 100 runs. Table 7.1 shows the results obtained by the different GO methods applied and Figure 7.3 presents the convergence curves for the best solution obtained by every method. For this problem, the *log_var* option was activated for all decision variables in *SSm* to generate the initial set of diverse solutions covering different orders of magnitude.

Solver	**Best**	**Mean**	**Worst**	**Mean Evaluations**	**Mean CPU time (s)**
CMAES	31113	31393	32945	10004	130
DE	348.56	22515	31951	10000	127
glcDirect	36218	-	-	9996	119
OQNLP	31252	-	-	10000	119
SRES	31251	32651	41864	10000	121
SSm	19.872	19.872	19.872	9518	122

Table 7.1: Results for the α-pinene isomerization problem

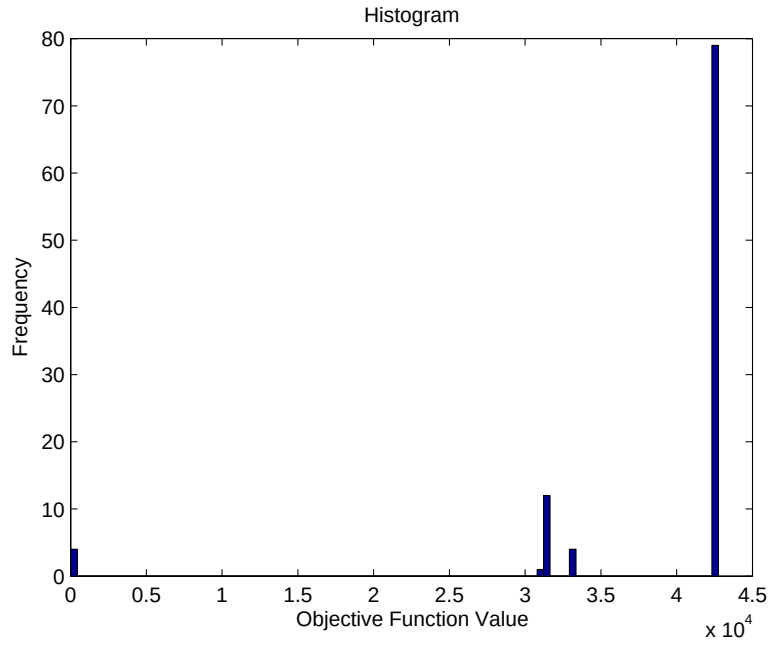


Figure 7.2: Histogram of solutions obtained from the multistart procedure using *n2fb* for the α-pinene isomerization problem

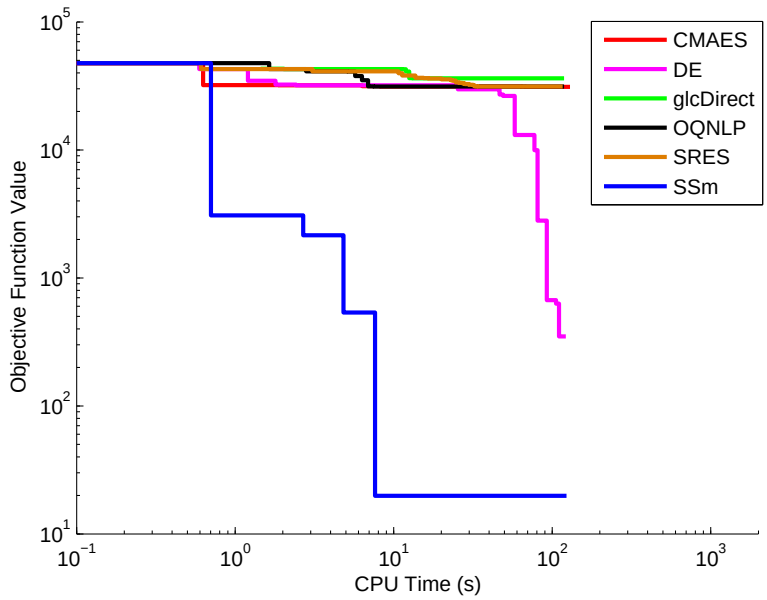


Figure 7.3: Convergence curves for the different solvers in the α-pinene isomerization problem

The results show that *SSm* was the only method that found the best known solution in the budget of function evaluations fixed. Actually, it found this solution in all the runs. Convergence curves show how *SSm* provided better values than the other solvers in less computation time. *DE* also provided a good value, but not as good as *SSm*'s best. The rest

of solvers got stuck in local minima. The only deterministic solver used in this comparison, *glcDirect*, presented the poorest results even if the dimension of the problem is rather small.

Figure 7.4 shows a comparison between the model predicted values and the experimental data reported by Fuguitt and Hawkins (1947) corresponding to the concentration of the reactant and the four products. The parameters estimated with *SSm* allow to reproduce almost exactly the experimental data.

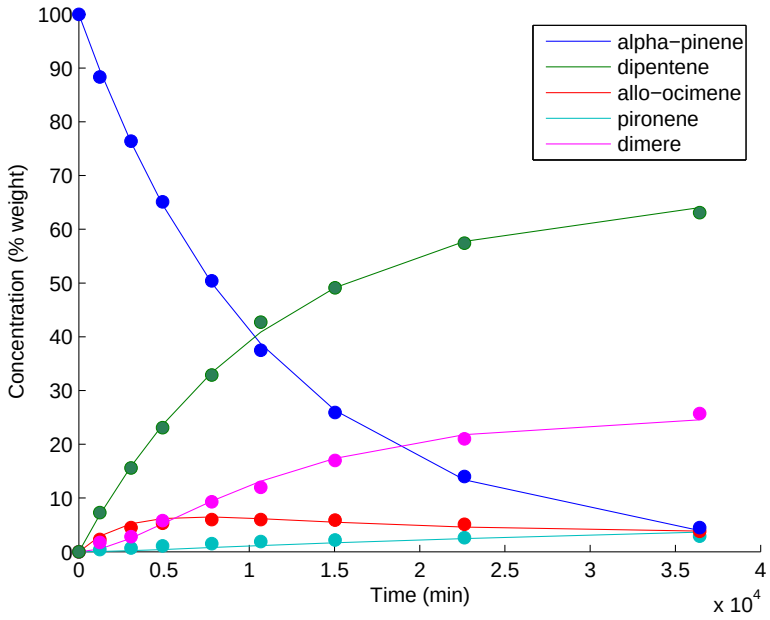


Figure 7.4: Experimental data vs. predicted values using the parameters estimated with *SSm*

7.2 Inhibition of HIV proteinase

7.2.1 Introduction

This problem deals with the estimation of a number of rate constants of a model for the reaction mechanism of irreversible inhibition of HIV proteinase, as originally studied by Kuzmic (1996). In Figure 7.5, the mechanism is depicted: the HIV proteinase (E) is added to a solution of an irreversible inhibitor (I) and a fluorogenic substrate (S). The enzyme is only active in a dimer form, the product is a competitive inhibitor for the substrate and the inhibitor is irreversible.

The problem considers an experiment where HIV proteinase (assay concentration 0.004 μM) was added to a solution of an irreversible inhibitor and a fluorogenic substrate (25

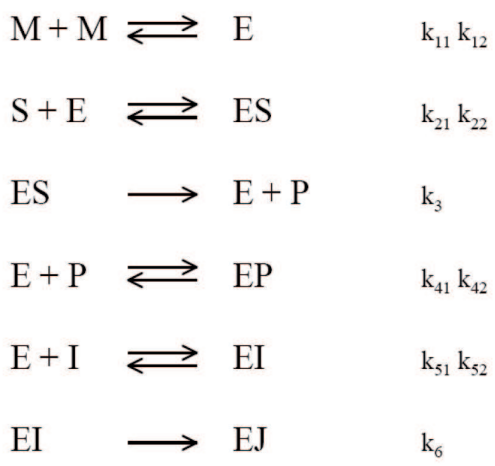


Figure 7.5: Mechanism of irreversible inhibition of HIV proteinase

μM). The fluorescence changes were monitored for one hour in each of the five experiments conducted at four different concentrations of the inhibitor (0, 0.0015, 0.003, and 0.004 μM in replicate).

We considered the same problem solved by Kuzmic (1996) and Mendes and Kell (1998), fitting five of the rate constants to the experimental data. In this fit, a certain degree of uncertainty ($\pm$ 50 %) in the value of the initial concentrations of substrate and enzyme (titration errors) was also assumed. In addition, the offset (baseline) of the fluorimeter was also considered as a degree of freedom. Given that there are five time course curves, there are a total of 20 adjustable parameters: the five rate constants, five initial concentrations of enzyme, five initial concentrations of substrate and five offset values.

The mathematical model consists of a set of 9 nonlinear ODE's with ten parameters. This can be described as follows:

$$\frac{d[M]}{dt} = -2k_{11}[M][M] + 2k_{12}[E] \tag{7.7}$$
$$\frac{d[P]}{dt} = k_3[ES] - k_{41}[P][E] + k_{42}[EP] \tag{7.8}$$
$$\frac{d[S]}{dt} = -k_{21}[S][E] + k_{22}[ES] \tag{7.9}$$
$$\frac{d[I]}{dt} = -k_{51}[I][E] + k_{52}[EI] \tag{7.10}$$
$$\frac{d[ES]}{dt} = k_{21}[S][E] - k_{22}[ES] - k_3[ES] \tag{7.11}$$
$$\frac{d[EP]}{dt} = k_{41}[P][E] - k_{42}[EP] \tag{7.12}$$

$$\frac{d[E]}{dt} = k_{11}[M][M] - k_{12}[E] - k_{21}[S][E] + k_{22}[ES] + k_3[ES] \quad (7.13)$$
$$-k_{41}[P][E] + k_{42}[EP] - k_{51}[I][E] + k_{52}[EI]$$
$$\frac{d[EI]}{dt} = k_{51}[I][E] - k_{52}[EI] - k_6[EI] \quad (7.14)$$
$$\frac{d[EJ]}{dt} = k_6[EI] \quad (7.15)$$

Like in the previous case, the cost function to be minimized is formulated as:

$$J(p) = \sum_{j=1}^{20} \sum_{i=1}^{5} (y_j(p, t_i) - \tilde{y}_{ji})^2 \quad (7.16)$$

By minimization of the sum-of-squares function of the residuals between the measured and the simulated data, the best known solution ($f(x) = 0.0211$) was obtained by Mendes and Kell (1998) using simulated annealing, with a computational cost of 3 million simulations. The next best solution ($f(x) = 0.0213$) was obtained using a Levenberg-Marquardt method in a considerable shorter computational time (4000 simulations) although this method is only guaranteed to converge to the global minimum if started in its vicinity.

7.2.2 Numerical results

The histogram depicting the multistart procedure to check the multimodality of the problem is shown in Figure 7.6. It shows the practical multimodality of the problem. The multistart procedure obtains function values near to the best known solution, improving the results obtained by Mendes and Kell (1998) (the best value reported is $f(x) = 0.019968$). The multistart procedure was able to find solutions with similar function values in 14 runs out of 100.

In this case, we used the same local solver (*n2fb* in double precision) in our algorithm *SSm*. The local search frequency was increased by setting `opts.local.n1=0` and `opts.local.n2=0` (see Appendix A for information about these parameters).

Table 7.2 shows the results obtained by the different GO methods applied and Figure 7.7 presents the convergence curves for the runs achieving the best solution obtained by every method. In Table 7.3, the values of the bounds and the best solution obtained by *SSm* are shown.

In this small budget of function evaluations, *SSm* is the only algorithm which achieves a better value than the one reported by Mendes and Kell (1998). In the rest of runs, the

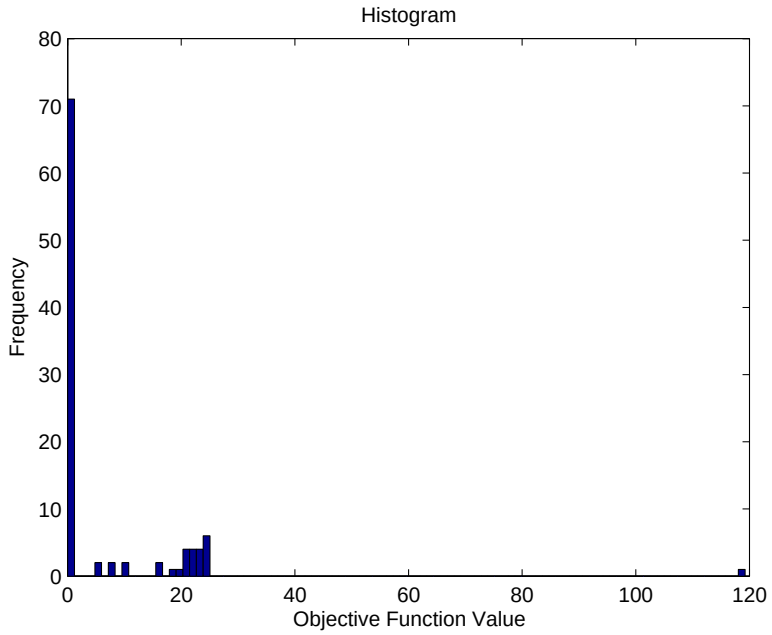


Figure 7.6: Histogram of solutions obtained from the multistart procedure using *n2fb* in double precision for the inhibition of HIV proteinase problem

Solver	Best	Mean	Worst	Mean Evaluations	Mean CPU time (s)
CMAES	0.49588	1.6014	3.4060	30017	1104
DE	0.32965	0.73351	1.05645	30000	844
glcDirect	10.490	-	-	33082	3004
OQNLP	1.4266	-	-	30000	2631
SRES	0.28157	0.35943	0.42693	30000	924
SSm	0.019764	0.020599	0.020825	29345	1294

Table 7.2: Results for the inhibition of HIV proteinase problem

solutions were of the same order. The rest of solvers provided solutions at least one order of magnitude higher.

Despite the fact that *SSm* converged in every run to solutions with very good values of the cost function, the values of the parameters were not always the same (see examples in Table 7.4) indicating a very flat objective function in the region of parameter space near the optimum. This indicates the lack of identifiability for this problem. This characteristic is reported and explained in Rodríguez-Fernández et al. (2006a). However, it is worth noting the very good correlation between the experimental and predicted data for the best decision vector (see Figure 7.8).

Parameter	Best *SSm* solution	Lower Bound	Upper Bound
k_3	6.66e+0	0.00e+0	1.00e+5
k_{42}	9.31e+4	0.00e+0	1.00e+5
k_{22}	6.34e+2	0.00e+0	1.00e+5
k_{52}	3.59e+4	0.00e+0	1.00e+5
k_6	3.68e+3	0.00e+0	1.00e+5
S_0 (exp 1)	2.47e+1	1.25e+1	3.75e+1
S_0 (exp 2)	2.35e+1	1.25e+1	3.75e+1
S_0 (exp 3)	2.71e+1	1.25e+1	3.75e+1
S_0 (exp 4)	1.58e+1	1.25e+1	3.75e+1
S_0 (exp 5)	1.40e+1	1.25e+1	3.75e+1
E_0 (exp 1)	5.50e-3	2.00e-3	6.00e-3
E_0 (exp 2)	5.33e-3	2.00e-3	6.00e-3
E_0 (exp 3)	6.00e-3	2.00e-3	6.00e-3
E_0 (exp 4)	4.35e-3	2.00e-3	6.00e-3
E_0 (exp 5)	3.99e-3	2.00e-3	6.00e-3
offset (exp 1)	-5.26e-3	-2.00e-1	4.00e-1
offset (exp 2)	-6.21e-3	-2.00e-1	4.00e-1
offset (exp 3)	-1.71e-2	-2.00e-1	4.00e-1
offset (exp 4)	-8.24e-3	-2.00e-1	4.00e-1
offset (exp 5)	3.15e-3	-2.00e-1	4.00e-1

Table 7.3: Bounds and best solution for the inhibition of HIV proteinase problem

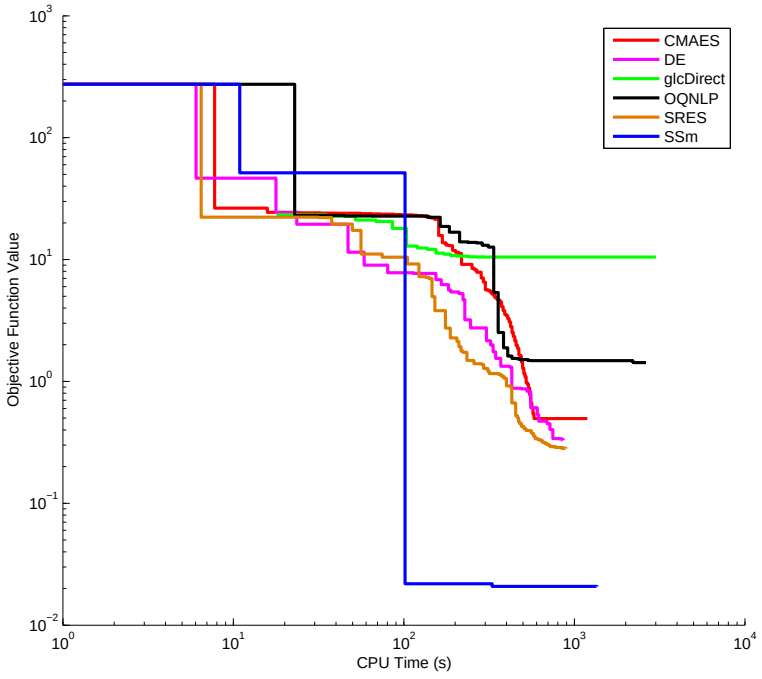


Figure 7.7: Convergence curves for the different solvers in the inhibition of HIV proteinase problem

Results *SSm*		
Parameter	J=0.019764	J=0.019967
$\boldsymbol{k_3}$	6.66e+0	5.53e+0
$\boldsymbol{k_{42}}$	9.31e+4	2.46e+2
$\boldsymbol{k_{22}}$	6.34e+2	4.63e+1
$\boldsymbol{k_{52}}$	3.59e+4	4.55e+2
$\boldsymbol{k_6}$	3.68e+3	6.24e+4
$\boldsymbol{S_0}$ **(exp 1)**	2.47e+1	2.47e+1
$\boldsymbol{S_0}$ **(exp 2)**	2.35e+1	2.35e+1
$\boldsymbol{S_0}$ **(exp 3)**	2.71e+1	2.71e+1
$\boldsymbol{S_0}$ **(exp 4)**	1.58e+1	1.67e+1
$\boldsymbol{S_0}$ **(exp 5)**	1.40e+1	1.41e+1
$\boldsymbol{E_0}$ **(exp 1)**	5.50e-3	5.38e-3
$\boldsymbol{E_0}$ **(exp 2)**	5.33e-3	5.17e-3
$\boldsymbol{E_0}$ **(exp 3)**	6.00e-3	6.00e-3
$\boldsymbol{E_0}$ **(exp 4)**	4.35e-3	4.25e-3
$\boldsymbol{E_0}$ **(exp 5)**	3.99e-3	3.97e-3
offset (exp 1)	-5.26e-3	-5.56e-3
offset (exp 2)	-6.21e-3	-5.31e-3
offset (exp 3)	-1.71e-2	-1.73e-2
offset (exp 4)	-8.24e-3	-1.11e-2
offset (exp 5)	3.15e-3	4.42e-4

Table 7.4: Parameters for two solutions in the inhibition of HIV proteinase problem

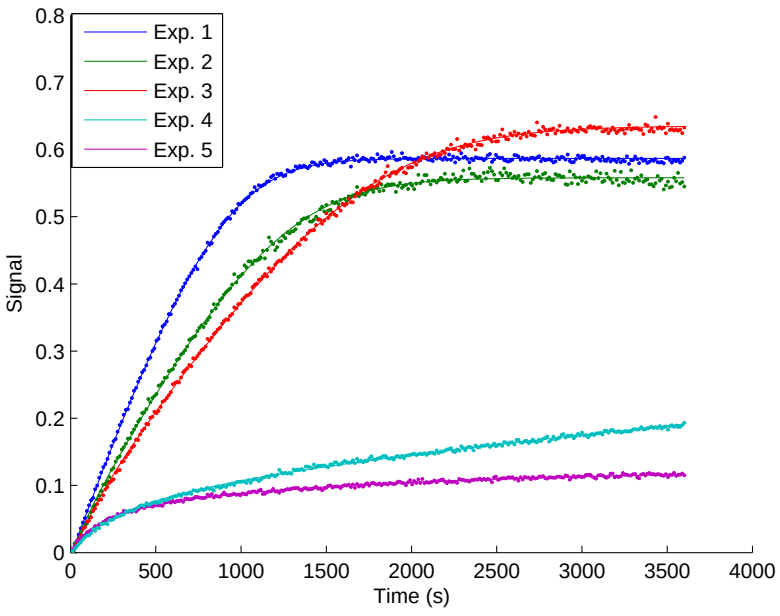


Figure 7.8: Experimental data vs. predicted values using the parameters estimated with *SSm*

7.3 Three-step biochemical pathway

7.3.1 Introduction

This case study, considered as a challenging benchmark problem by Moles et al. (2003b), involves a biochemical pathway with three enzymatic steps, including the enzymes and mRNAs

explicitly. Figure 7.9 contains a diagram illustrating the network of reactions and kinetics effects (feedback loops).

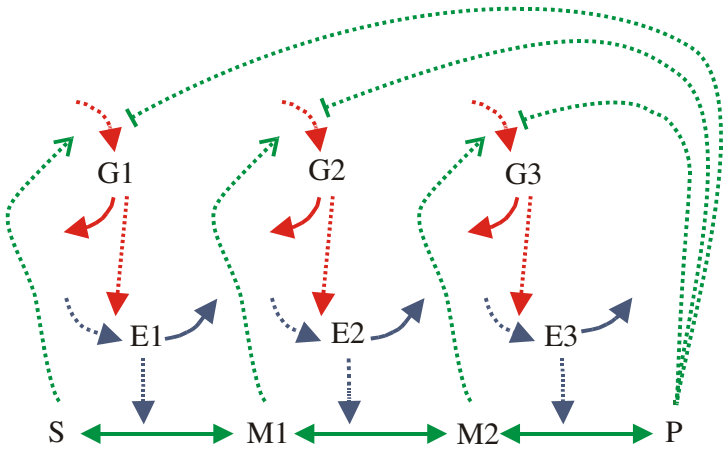


Figure 7.9: Three-step biochemical pathway scheme

The identification problem consists of the estimation of 36 kinetic parameters of the nonlinear biochemical dynamic model (8 nonlinear ODEs) which describes the variation of the metabolite concentration over time.

The mathematical formulation of this dynamic nonlinear model is described by:

$$\frac{dG_1}{dt} = \frac{V_1}{1 + \left(\frac{P}{Ki_1}\right)^{ni_1} + \left(\frac{Ka_1}{S}\right)^{na_1}} - k_1\ G_1 \tag{7.17}$$

$$\frac{dG_2}{dt} = \frac{V_2}{1 + \left(\frac{P}{Ki_2}\right)^{ni_2} + \left(\frac{Ka_2}{M_1}\right)^{na_2}} - k_2\ G_2 \tag{7.18}$$

$$\frac{dG_3}{dt} = \frac{V_3}{1 + \left(\frac{P}{Ki_3}\right)^{ni_3} + \left(\frac{Ka_3}{M_2}\right)^{na_3}} - k_3\ G_3 \tag{7.19}$$

$$\frac{dE_1}{dt} = \frac{V_4\ G_1}{K_4 + G_1} - k_4\ E_1 \tag{7.20}$$

$$\frac{dE_2}{dt} = \frac{V_5\ G_2}{K_5 + G_2} - k_5\ E_2 \tag{7.21}$$

$$\frac{dE_3}{dt} = \frac{V_6\ G_3}{K_6 + G_3} - k_6\ E_3 \tag{7.22}$$

$$\frac{dM_1}{dt} = \frac{kcat_1\ E_1\ \left(\frac{1}{Km_1}\right)(S - M_1)}{1 + \frac{S}{Km_1} + \frac{M_1}{Km_2}} - \frac{kcat_2\ E_2\ \left(\frac{1}{Km_3}\right)(M_1 - M_2)}{1 + \frac{M_1}{Km_3} + \frac{M_2}{Km_4}} \tag{7.23}$$

$$\frac{dM_2}{dt} = \frac{kcat_2\ E_2\ \left(\frac{1}{Km_3}\right)(M_1 - M_2)}{1 + \frac{M_1}{Km_3} + \frac{M_2}{Km_4}} - \frac{kcat_3\ E_3\ \left(\frac{1}{Km_5}\right)(M_2 - P)}{1 + \frac{M_2}{Km_5} + \frac{P}{Km_6}} \tag{7.24}$$

where M_1, M_2, E_1, E_2, E_3, G_1, G_2 y G_3 represent the concentration of the 8 implied species in the different biochemical reactions. The cost function to be minimized is:

$$J = \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}((y_{teor}(i) - y_{exp}(i))_j)^2 \tag{7.25}$$

where m is the number of experiments and n is the number of data per experiment ($m = 16$ and $n = 20$ for this problem). w_{ij} are the weights to normalize the contribution of each term. They were calculated as $w_{ij} = 1/(\max(y_{exp}(i))_j)$. Substrate and product concentration (S and P, respectively) act as control variables and they are considered constants for every experiment since their concentrations are very high compared to the other species. They have fixed initial values for every experiment as shown in Table 7.5.

Exp.	S conc.	P conc.	Exp.	S conc.	P conc.
1	0.1	0.05	**9**	2.1544	0.05
2	0.1	0.13572	**10**	2.1544	0.13572
3	0.1	0.36840	**11**	2.1544	0.36840
4	0.1	1.0	**12**	2.1544	1.0
5	0.46416	0.05	**13**	10	0.05
6	0.46416	0.13572	**14**	10	0.13572
7	0.46416	0.36840	**15**	10	0.3684
8	0.46416	1.0	**16**	10	1.0

Table 7.5: S and P concentration values for all the experiments

All the necessary data to implement this problem (including the experimental data) can be found in the web page maintained by Julio R. Banga [1]. Moles et al. (2003b) tried to solve this problem using several deterministic and stochastic global optimization algorithms. They found that only a certain type of stochastic algorithms, evolution strategies (implemented as

[1] http://www.iim.csic.es/∼julio/GR03_statement.txt

the *SRES* code), was able to successfully solve it, although at a rather large computational cost. Rodríguez-Fernández et al. (2006b) presented a two-phase hybrid method which converged to better solutions, with speeds higher than more than one order of magnitude with respect to the previous results. Making use of our proposed algorithm, *SSm*, those results have been improved with computation time savings of two orders of magnitude, as shown in the next section.

7.3.2 Numerical results

The histogram resulting from the multistart procedure applied to this problem is shown in Figure 7.10. The best solution found by the multistart procedure ($f = 10.17$) is very far from the global optimum, which is 0.000 for this problem.

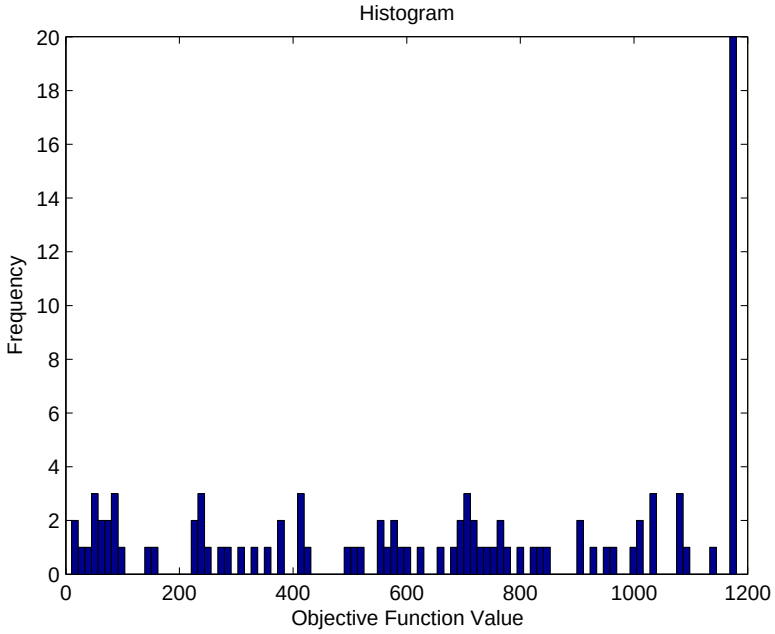


Figure 7.10: Histogram of solutions obtained from the multistart procedure using *n2fb* for the three-step biochemical pathway problem

This is a very difficult problem as demonstrated by the previous studies published. The global solution has been found after a very long computation time. A number of reasons make this problem so difficult, such as the presence of large flat areas, many local minima near the global one, a very narrow area of the basin of attraction of the global optimum compared to the search space and high function values very close to it, which make the algorithms neglect this area when searching around. For these reasons, we tuned our algorithm to make it more efficient. Instead of using the Euclidean distance to prevent stagnation, we used the

tolerance-based criterion (i.e., a solution must be different in all dimensions with respect to the *RefSet* members, using a relative tolerance, to join it). The relative tolerance fixed was 10^{-2}. Besides, the linear combinations proved to be more efficient for this problem than the hyper-rectangles based. These two modifications from the default parameters make the search more aggressive because less solutions apply to enter the *RefSet*. Thus, it is regenerated more often. Again, a specific local search algorithm for parameter estimation, *n2fb*, was used as *Improvement Method*. The final refinement was done with the same algorithm in double precision.

Table 7.6 shows the results obtained by the different GO methods applied and Figure 7.11 presents the convergence curves for the runs achieving the best solution obtained by every method. In Table 7.7, the values of the bounds and the best solution obtained by *SSm* are shown.

Solver	**Best**	**Mean**	**Worst**	**Mean Evaluations**	**Mean CPU time (s)**
CMAES	211.6	599.7	863.5	40019	258
DE	274.4	399.7	478.1	20160	297
glcDirect	328.8	-	-	42660	302
OQNLP	54.05	-	-	20160	69
SRES	290.7	406.3	513.8	20160	220
SSm	0.000	0.001	0.014	17454	180

Table 7.6: Results for the three-step biochemical pathway problem

Again, *SSm* is the only algorithm able to locate the global optimum in the budget of function evaluations fixed. Its results clearly improve previous results published for this problem. It finds the global solution reducing the computation time needed in three and two orders of magnitude compared to the results of Moles et al. (2003b) and Rodríguez-Fernández et al. (2006b), respectively, as it is illustrated in Figure 7.12.

Figure 7.13 shows a comparison between the predicted and experimental data for the 10th experiment, evidencing the accuracy of the fit. The representation of the dynamic behavior for the other experiments is quite similar and is not included here for the sake of brevity.

7.4 Conclusions

The results of the application of the different global optimization algorithms over the set of parameter estimation problems considered demonstrate the superiority of our proposed methodology to solve this type of nonlinear dynamic problems. In all cases, our method

Parameter	Best *SSm* solution	Lower Bound	Upper Bound
V_1	1	1e-12	1e+3
Ki_1	1	1e-12	1e+3
ni_1	2	1e-1	1e+1
Ka_1	1	1e-12	1e+3
na_1	2	1e-1	1e+1
k_1	1	1e-12	1e+3
V_2	1	1e-12	1e+3
Ki_2	1	1e-12	1e+3
ni_2	2	1e-1	1e+1
Ka_2	1	1e-12	1e+3
na_2	2	1e-1	1e+1
k_2	1	1e-12	1e+3
V_3	1	1e-12	1e+3
Ki_3	1	1e-12	1e+3
ni_3	2	1e-1	1e+1
Ka_3	1	1e-12	1e+3
na_3	2	1e-1	1e+1
k_3	1	1e-12	1e+3
V_4	0.1	1e-12	1e+3
K_4	1	1e-12	1e+3
k_4	0.1	1e-12	1e+3
V_5	0.1	1e-12	1e+3
K_5	1	1e-12	1e+3
k_5	0.1	1e-12	1e+3
V_6	0.1	1e-12	1e+3
K_6	1	1e-12	1e+3
k_6	0.1	1e-12	1e+3
$kcat_1$	1	1e-12	1e+3
Km_1	1	1e-12	1e+3
Km_2	1	1e-12	1e+3
$kcat_2$	1	1e-12	1e+3
Km_3	1	1e-12	1e+3
Km_4	1	1e-12	1e+3
$kcat_3$	1	1e-12	1e+3
Km_5	1	1e-12	1e+3
Km_6	1	1e-12	1e+3

Table 7.7: Bounds and best solution for the three-step biochemical pathway problem

found the best known solutions and presented the fastest convergence rate (in some cases improving the convergence rate by several orders of magnitude). The fact that our methodology combines a local search (i.e., the *Improvement Method*) with a global evolutionary phase helps to accelerate the convergence to the best known solutions. The use of a specific local algorithm for parameter estimation problems such as *n2fb* makes *SSm* a powerful tool to solve this kind of problems. Further, the different options implemented make the algorithm robust enough to handle problems of different types and levels of difficulty.

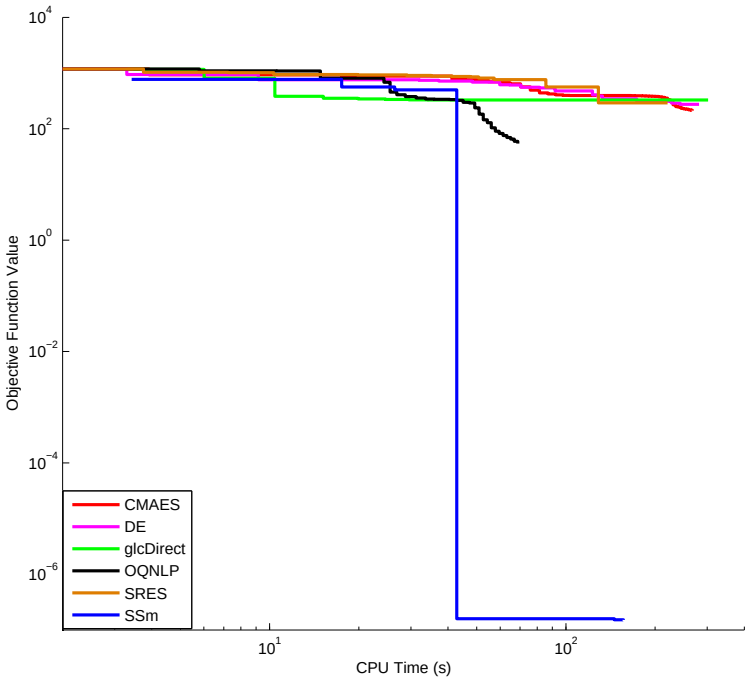


Figure 7.11: Convergence curves for the different solvers in the three-step biochemical pathway problem

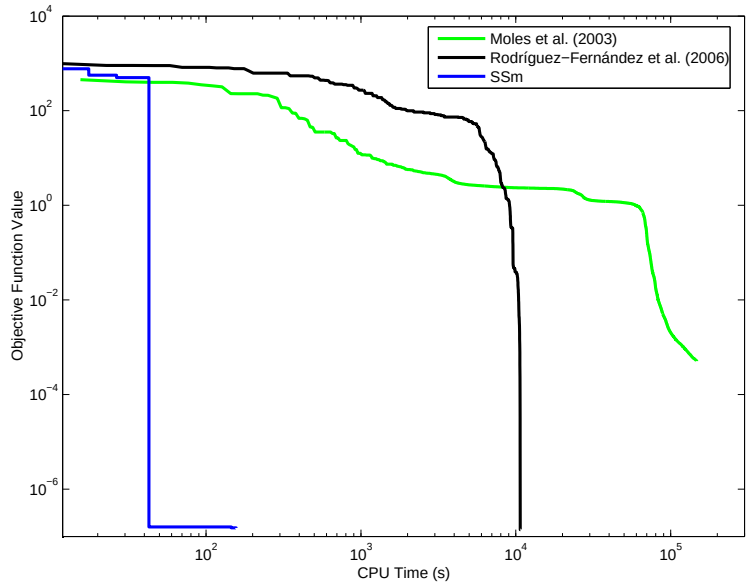


Figure 7.12: Comparison of convergence curves for the three-step biochemical pathway problem

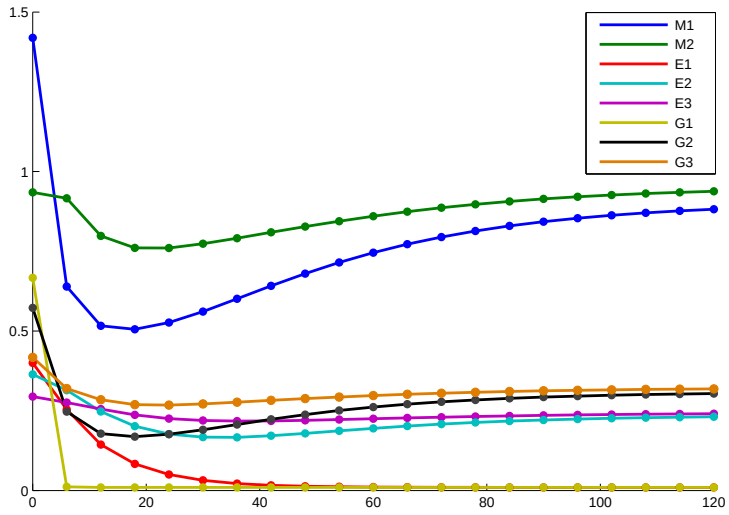


Figure 7.13: Experimental data vs. predicted values in one experiment using the parameters estimated with *SSm*

Chapter 8

Integrated design and control problems

8.1 Introduction

The problem of integrated design and control optimization of process plants is discussed in this chapter. We consider it as a nonlinear programming problem subject to differential-algebraic constraints (see Section 1.1.2). This class of problems is frequently multimodal and "costly" (i.e., computationally expensive to evaluate). Two challenging Wastewater Treatment Plants (WWTPs) benchmark models are used here to evaluate the performance of the optimization techniques proposed in this work.

The simultaneous (integrated) bioprocess design approach, considering operability, control and economic issues, has been widely recognized in recent years. The aim is to obtain profitable and operable process and control structures in a systematic way. Both the process design characteristics, control strategies, control structure and controller's tuning parameters have to be selected optimally in order to minimize the total cost of the system while satisfying a large number of feasibility constraints in the presence of time-varying disturbances. Consequently, the use of global optimization techniques is strongly advisable. Recent contributions dealing with the integrated design of WWTPs can be found in Rigopoulos and Linke (2002) and Alasino et al. (2007).

Biochemical processes are difficult to control due to the sensitivity of the microorganisms and the lack of full knowledge to control intracellular processes by modifying the external

conditions (Bogle et al., 1996). A number of control strategies have been proposed to meet the strict standards that WWTPs must comply with, while also trying to reduce costs. Relevant examples from the recent literature of attempts to optimize the controllers of these plants are:

- *ad hoc* extensive simulation studies (Ladiges et al., 1999; Carucci et al., 1999; Schütze et al., 2002; Ladiges and Günner, 2003; Cappai et al., 2004; Butler and Schütze, 2005). Strictly speaking these may not be called optimizations, because there is no evidence that a locally or globally best solution is found.

- Dynamic optimization design strategies using local gradient-based (Chachuat et al., 2001) or evolutionary (Balku and Berber, 2006) optimization methods, often based on simplified models and without use of any control strategy.

- Global optimization methods for multiobjective optimal control of wastewater systems (Moles et al., 2003a; Sendín et al., 2004; Fu et al., 2008).

- An integrated approach for the optimization of control strategies, where a small selection of global and local optimization methods was used (Schütze. et al., 1999; Schütze et al., 2001).

Evaluation of these and similar strategies, either in practice or by simulation, is a real problem due to the lack of a standard with respect to evaluation criteria, process complexity and large variations in plant configuration.

8.2 Problem WWTP1: Simultaneous design and control of a WWT plant

8.2.1 Introduction

This case study represents an alternative configuration of a real wastewater treatment plant placed in Manresa (Spain), as described by Gutiérrez and Vega (2000). The plant is formed by two aeration tanks, acting as bioreactors, and two settlers (see Figure 8.1).

A flocculating microbial population (biomass) is kept inside each bioreactor, transforming the biodegradable pollutants (substrate), with the aeration turbines providing the necessary level of dissolved oxygen. The effluents from the aeration tanks are separated in their associated settlers into a clean water stream and activated sludge, which is recycled to the

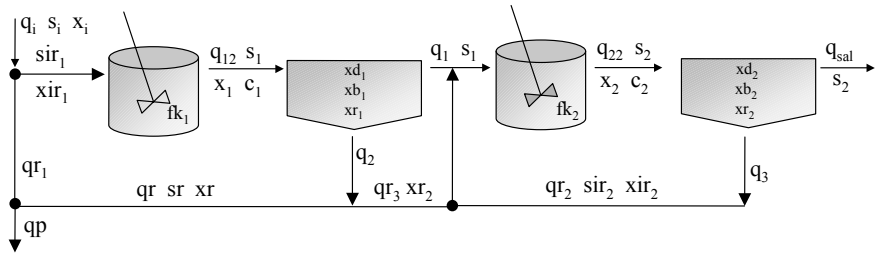


Figure 8.1: WWT plant scheme

corresponding aeration tank. Since the activated sludge is constantly growing, more is produced that can be recycled to the tanks, so the excess is eliminated via a purge stream (q_p). The objective of the control system is to keep the substrate concentration at the output (s_2) under a given admissible value. The main disturbances come from large variations in both the flowrate and substrate concentration (q_i and s_i) of the input stream. Although there are several possibilities for the manipulated variable, here we have considered the flowrate of the sludge recycle to the first aeration tank, as considered by Gutiérrez and Vega (2000).

The process dynamic model is derived from a first principles approach. The overall model consists of 33 DAEs (14 of them are ODEs) and 44 variables. The value of three flowrates (qr_2, qr_3 and qp) are fixed at their steady-state values corresponding to a certain nominal operational conditions. Therefore, this leaves 8 design variables for the integrated design problem, namely v_1, v_2 (volume of the aeration tanks, m^3), ad_1, ad_2 (areas of the settlers, m^2), fk_1, fk_2 (aeration factors) and k_p, τ_i (the proportional gain and the integral time of the PI controller, respectively). More complex formulations are possible, but our objective is to illustrate how this problem of medium size is already very challenging for many GO methods.

The nonlinear mathematical model arises from the mass balances (8.1-8.6) in the aerated tanks. They represent the changes in the oxygen concentration (c_1, c_2), biomass (x_1, x_2) and organic substrate. Similarly, equations 8.7-8.12 represent the mass balances in the settlers, where three different and increasing levels of biomass concentration are considered (xd_1, xb_1, xr_1 and xd_2, xb_2, xr_2). Finally, equations 8.13 and 8.14 describe the integral term of the controller and the *ISE* (Integral Squared Error).

$$\frac{dx_1}{dt} = \frac{yy \cdot \mu \cdot s_1 \cdot x_1}{ks + s_1} - kc \cdot x_1 - \frac{kd \cdot x_1^2}{s_1} + \frac{q_{12} \cdot (xir_1 - x_1)}{v_1} \tag{8.1}$$

$$\frac{dx_2}{dt} = \frac{yy \cdot \mu \cdot s_2 \cdot x_2}{ks + s_2} - kc \cdot x_2 - \frac{kd \cdot x_2^2}{s_2} + \frac{q_{22} \cdot (xir_2 - x_2)}{v_2} \tag{8.2}$$

$$\frac{ds_1}{dt} = -\frac{\mu \cdot s_1 \cdot x_1}{ks + s_1} + fkd \cdot (\frac{kd \cdot x_1^2}{s_1} + kc \cdot x_1) + \frac{q_{12} \cdot (sir_1 - s_1)}{v_1} \tag{8.3}$$

$$\frac{ds_2}{dt} = -\frac{\mu \cdot s_2 \cdot x_2}{ks + s_2} + fkd \cdot (\frac{kd \cdot x_2^2}{s_2} + kc \cdot x_2) + \frac{q_{22} \cdot (sir_2 - s_2)}{v_2} \tag{8.4}$$

$$\frac{dc_1}{dt} = kla \cdot fk_1 \cdot (c_s - c_1) - \frac{k_{01} \cdot \mu \cdot x_1 \cdot s_1}{ks + s_1} - \frac{q_{12} \cdot c_1}{v_1} \tag{8.5}$$

$$\frac{dc_2}{dt} = kla \cdot fk_2 \cdot (c_s - c_2) - \frac{k_{01} \cdot \mu \cdot x_2 \cdot s_2}{ks + s_2} + \frac{q_1 \cdot c_1}{v_2} - \frac{q_{22} \cdot c_2}{v_2} \tag{8.6}$$

$$\frac{dxd_1}{dt} = \frac{(q_{12} - q_2) \cdot xb_1 - q_1 \cdot xd_1}{ad_1 \cdot ld_1} - \frac{vsd_1}{ld_1} \tag{8.7}$$

$$\frac{dxb_1}{dt} = \frac{q_{12} \cdot x_1 - q_1 \cdot xb_1 - q_2 \cdot xb_1}{ad_1 \cdot lb_1} + \frac{vsd_1 - vsb_1}{lb_1} \tag{8.8}$$

$$\frac{dxd_2}{dt} = \frac{(q_{22} - q_3) \cdot xb_2 - qsal \cdot xd_2}{ad_2 \cdot ld_2} - \frac{vsd_2}{ld_2} \tag{8.9}$$

$$\frac{dxb_2}{dt} = \frac{q_{22} \cdot x_2 - q_{sal} \cdot xb_2 - q_3 \cdot xb_2}{ad_2 \cdot lb_2} + \frac{vsd_2 - vsb_2}{lb_2} \tag{8.10}$$

$$\frac{dxr_1}{dt} = \frac{q_2 \cdot (xb_1 - xr_1)}{ad_1.lr_1} + \frac{vsb_1}{lr_1} \tag{8.11}$$

$$\frac{dxr_2}{dt} = \frac{q_3 \cdot (xb_2 - xr_2)}{ad_2 \cdot lr_2} + \frac{vsb_2}{lr_2} \tag{8.12}$$

$$\frac{dI}{dt} = \frac{k_p}{\tau_i} \cdot (s_{2s} - s_2) \tag{8.13}$$

$$\frac{d(ISE)}{dt} = (s_{2s} - s_2) \cdot (s_{2s} - s_2) \tag{8.14}$$

The algebraic equation 8.15 corresponds to the control law (qr_{1s} is its stationary value), whereas equations 8.18-8.21 specify the settling velocity of the sludge. Equations 8.22-8.28 are the balances among the system flow rates (m^3/h). Equation 8.33 represents the perturbation to the inlet (substrate) considered in the *ISE* calculation. This perturbation is introduced 25 hours after the plant is working in steady state.

$$qr_1 = qr_{1s} + k_p \cdot (s_{2s} - s_2) + I \tag{8.15}$$

$$sr_1 = \frac{s_1 \cdot q_2 + qr_3 \cdot s_2}{q_r} \tag{8.16}$$

$$x_r = \frac{xr_1 \cdot q_2 + xr_2 \cdot qr_3}{q_r} \tag{8.17}$$

$$vsd_1 = nnr \cdot xd_1 \cdot e^{aar \cdot xd_1} \tag{8.18}$$

$$vsb_1 = nnr \cdot xb_1 \cdot e^{aar \cdot xb_1} \tag{8.19}$$

$$vsd_2 = nnr \cdot xd_2 \cdot e^{aar \cdot xd_2} \tag{8.20}$$

$$vsb_2 = nnr \cdot xb_2 \cdot e^{aar \cdot xb_2} \tag{8.21}$$

$$q_2 = qr_1 + q_p - qr_3 \tag{8.22}$$

$$q_3 = qr_3 + qr_2 \tag{8.23}$$

$$q_{12} = q_i + qr_1 \tag{8.24}$$

$$q_{22} = q_1 + qr_2 \tag{8.25}$$

$$q_{sal} = q_i - q_p \tag{8.26}$$

$$q_1 = q_{12} - q_2 \tag{8.27}$$

$$q_r = q_2 + qr_3 \tag{8.28}$$

$$xir_1 = \frac{q_i \cdot x_i + qr_1 \cdot x_r}{q_{12}} \tag{8.29}$$

$$sir_1 = \frac{q_i \cdot s_i + qr_1 \cdot sr_1}{q_{12}} \tag{8.30}$$

$$xir_2 = \frac{q_1 \cdot xd_1 + xr_2 \cdot qr_2}{q_{22}} \tag{8.31}$$

$$sir_2 = \frac{q_1 \cdot s_1 + s_2 \cdot qr_2}{q_{22}} \tag{8.32}$$

$$s_i = \left\{ \begin{array}{ll} s_{i,s} & t < 25h \\ s_{i,s} + (10 - 10 \cdot e^{-2.5t}) & t \geq 25h \end{array} \right\} \tag{8.33}$$

Apart from the DAEs, which act as equality constraints, the optimization problem is subject to a set of inequality constraints referring to residence time and other relations which should be within some defined limits:

$$2.5 \leq \frac{v_1}{q_{12}} \leq 8.0 \tag{8.34}$$

$$0.001 \leq \frac{q_i \cdot s_i + qr_1 \cdot sr_1}{v_1 \cdot x_1} \leq 0.6 \tag{8.35}$$

$$0.001 \leq \frac{(q_i + qr_3 - q_p) \cdot s_1 + qr_2 \cdot s_2}{v_2 \cdot x_2} \leq 0.06 \tag{8.36}$$

$$\frac{q_{12}}{ad_1} \leq 1.5 \tag{8.37}$$

$$\frac{q_{22}}{ad_2} \leq 1.5 \tag{8.38}$$

$$3.0 \leq \frac{v_1 \cdot x_1 + ad_1 \cdot lr_1 \cdot xr_1}{q_p \cdot xr_1 \cdot 24} \leq 10.0 \tag{8.39}$$

$$3.0 \leq \frac{v_2 \cdot x_2 + ad_2 \cdot lr_2 \cdot xr_2}{q_p \cdot xr_2 \cdot 24} \leq 10.0 \tag{8.40}$$

$$0.5 \leq \frac{q_2 + q_3}{q_i} \leq 0.9 \tag{8.41}$$

$$0.03 \leq \frac{q_p}{q_2 + q_3} \leq 0.07 \tag{8.42}$$

Additionally, there is a set of 30 double inequality constraints which define valid ranges

for the state variables (see Table 8.1). All these constraints must be checked before and after introducing the perturbation, giving a total number of 120 inequality constraints.

var	lb	ub	var	lb	ub	var	lb	ub
x_1	500	3000	xb_2	30	3000	xir_2	200	2000
x_2	200	3000	xr_2	1000	10000	sir_2	30	500
s_1	25	300	sr_1	20	1000	q_2	200	3000
s_2	20	125	xr	2000	8750	q_3	200	3000
c_1	1	8	vsd_1	100	2000	q_{12}	50	3500
c_2	1	8	vsb_1	300	3000	q_{22}	50	3500
xd_1	10	300	vsd_2	10	2000	q_{sal}	100	3000
xb_1	50	3000	vsb_2	100	3000	q_1	50	3000
xr_1	3000	10000	xir_1	400	2500	q_r	50	2000
xd_2	3	300	sir_1	50	100	qr_1	50	3000

Table 8.1: Bounds for the inequality constraints for the state variables

The integrated design problem is formulated as an NLP-DAEs, where the objective function to be minimized is a weighted sum of economic (ϕ_{econ}) and controllability cost terms (measured here as the ISE):

$$\begin{aligned} C = w_1 \cdot ISE + \phi_{econ} = w_1 \cdot ISE + (w_2 \cdot v_1^2) + \\ (w_3 \cdot v_2^2) + (w_4 \cdot ad_1^2) + (w_5 \cdot ad_2^2) + \\ (w_6 \cdot fk_1^2) + (w_7 \cdot fk_2^2) \end{aligned} \tag{8.43}$$

where the ISE is the integral square error, $ISE = \int_0^\infty e^2(t) \cdot dt$. The ISE is evaluated considering a step disturbance to the input substrate concentration, s_i, whose behavior is taken from the real plant (similarly to Schweiger and Floudas 1997). The minimization is subject to several sets of constraints.

- The 33 model DAEs (system dynamics), acting as differential-algebraic equality constraints.

- 32 inequality constraints which impose limits on the residence times and biomass load in the aeration tanks, the hydraulic capacity in the settlers, the sludge ages in the decanters, and the recycle and purge flow rates, respectively.

- An additional set of 120 double inequality constraints on the state variables (see Table 8.1).

A weighting vector $w_i = [10^3, 2 \cdot 10^{-5}, 2 \cdot 10^{-5}, 1 \cdot 10^{-5}, 1 \cdot 10^{-5}, 12, 12]$ was considered for the optimization runs, which implies a similar contribution of each term in the objective function.

8.2.2 Numerical results

The histogram depicting the multistart procedure using a SQP method (*fmincon*) to check the multimodality of the problem is shown in Figure 8.2. Only solutions with function values lower than 10000 are plotted in the histogram.

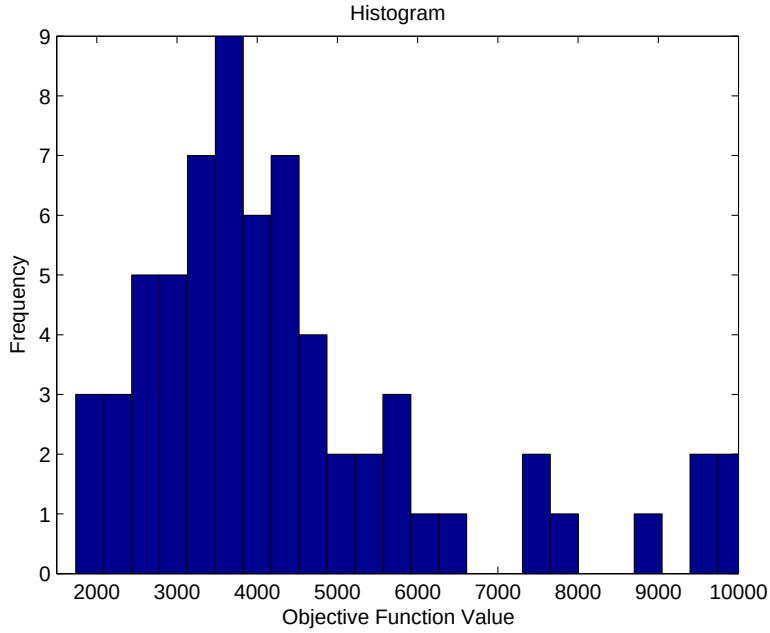


Figure 8.2: Histogram of solutions obtained from the multistart procedure using *fmincon* for problem WWTP1

The histogram shows the practical non-convexity of the problem and the best value reported ($f(x) = 1738.7$) is far from the best known solutions (around 1538) reported by Moles et al. (2003a) and Egea et al. (2007a). The *Improvement Method* was deactivated in *SSm* because it consumes excessive running time without significant solution improvement. However, a final refinement phase was activated using the direct search solver *fminsearch*. The reason for these special settings is the presence of discontinuities in the problem, which makes local algorithms fail or converge prematurely. Due to some execution errors, the local search was also deactivated in *OQNLP*. Egea et al. (2007a) already reported better results deactivating the local search for this problem.

Table 8.2 shows the results obtained by the different GO methods applied and Figure 8.3

presents the convergence curves for the runs achieving the best solution obtained by every method. The initial point used for this problem was chosen to obtain a feasible solution, and it is reported together with the bounds and the best found vector in Table 8.3.

Solver	Best	Mean	Worst	Mean Evaluations	Mean CPU time (s)
CMAES	1537.8	1540.7	1551.4	15002	357
DE	1537.8	1537.8	1537.8	15040	364
glcDirect	2201.8	-	-	15988	138
OQNLP	1663.6	-	-	20000	373
SRES	1537.8	1538.0	1539.0	15040	335
SSm	1537.8	1538.2	1539.0	15006	276

Table 8.2: Results for problem WWTP1

Parameter	Best *SSm* solution	Lower Bound	Upper Bound	Initial point
v_1	5493.01	1500	10000	8843.95
v_2	3982.54	1500	10000	7520.32
ad_1	2295.62	1000	4000	3994.72
ad_2	4000	1000	4000	3447.27
fk_1	0.0677195	0	1	0.7822
fk_2	0.0118226	0	1	0.7636
k_p	-99.9992	-100	-0.005	-9.507
τ_i	0.732842	0.5	100	10.7606
J	1537.8			13746

Table 8.3: Bounds, initial point and best *SSm* solution for problem WWTP1

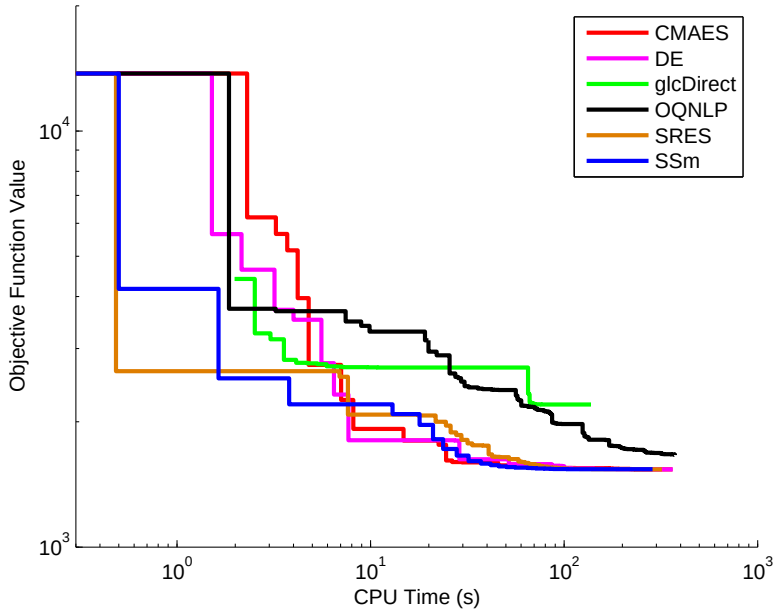


Figure 8.3: Convergence curves for the different solvers in problem WWTP1

Results for this problem show that many optimization methods present similar behavior in both, best solution found and convergence rate. *DE* presents the most consistent values, with no dispersion among the 10 runs. The deterministic method, *glcDirect*, provides the poorest result in the budget of function evaluations fixed.

To illustrate the improvements achieved by applying global optimization to the integrated design and control of the plant, a comparison between both indexes contained in the objective function before and after the optimization are shown in Table 8.4. Both the *ISE* and the economic term are clearly improved with respect to the initial operating point.

Index	**Value with x_0**	**Value with x_{SSm}^{best}**
$\boldsymbol{C}$	13745.6	1537.8
ISE	10.7574	0.4044
ϕ_{econ}	2988.16	1133.43

Table 8.4: Initial and optimized indexes for problem WWTP1

8.3 Problem WWTP-COST: a computationally expensive model

8.3.1 Introduction

In order to enhance the development and acceptance of new control strategies, the International Water Association (IWA) Task Group on Respirometry, together with the European COST work group, proposed a standard simulation benchmarking methodology for evaluating the performance of activated sludge plants. The COST 624 work group defines the benchmark as *a protocol to obtain a measure of performance of control strategies for activated sludge plants based on numerical, realistic simulations of the controlled plant.* According to this definition, the benchmark consists of a description of the plant layout, a simulation model and definitions of controller performance criteria. The layout of this benchmark plant combines nitrification with predenitrification by a five compartment reactor with an anoxic zone (see Figure 8.4). A secondary settler composed by 10 layers separates the microbial culture from the liquid being treated. A basic control strategy consisting of 2 PI controllers is proposed to test the benchmark. Its aim is to control the dissolved oxygen level in the final compartment of the reactor (AS Unit 5) by manipulation of the oxygen transfer, and to control the nitrate level in the last anoxic compartment (AS Unit 2) by manipulating the internal recycle flow rate (Jeppsson, 1996).

In this work, a Simulink implementation of the benchmark model by Jeppsson was used

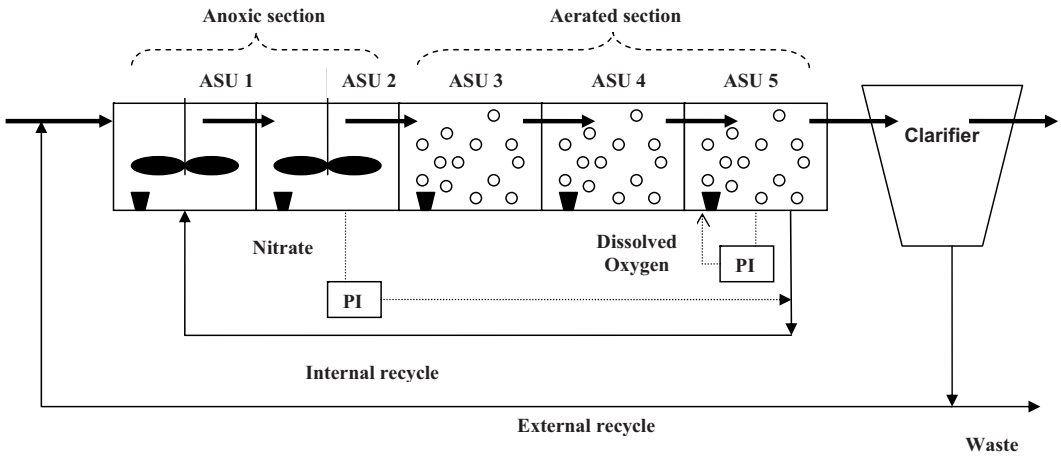


Figure 8.4: WWT COST plant scheme

for the simulations (for more information about the implementation, see Alex et al. 1999; Copp 2001; Jeppsson and Pons 2004). Each function evaluation consists in an initialization period of 100 days to achieve steady state, followed by a period of 14 days of dry weather and a third period of 14 days of rainy weather. Calculations of the controllers performance criterion are based on data from the last 7 rain days.

Since each simulation of this benchmark model takes a significant time on a standard PC (about 90 seconds in a PC-PIV 2,4 GHz), it is an illustrating example to evaluate the surrogate-based optimization algorithms such as *rbfSolve*, *ego* (see Section 6.1) and *SSKm* (see Chapter 5).

The system dynamics is described by algebraic mass balance equations, ordinary differential equations for the biological processes in the bioreactors as defined by the ASM1-model (Henze et al., 1987), and the double-exponential settling velocity function (Takács et al., 1991) as a fair presentation of the settling process. The overall process is formed by 8 subprocesses and is described by a set of more than 100 DAE's with 13 state variables. For the sake of brevity, the detailed model of the full plant and the parameters and design variables values are not shown but they can be found in the *IWA Task Group on Benchmarking of Control Strategies for WWTPs* web page[1].

Given the physical design and the control strategy of the plant, there is a number of operating variables over which we can apply optimization techniques to minimize a performance index of the plant. In this work we have considered the variables listed in Table 8.5. Default

[1]http://www.ensic.inpl-nancy.fr/benchmarkWWTP/Bsm1/Benchmark1.htm

values are proposed by the benchmark authors.

Variable	Description	Symbol	Default value	Units
v_1	Proportional gain O_2 controller	$K_{(O)}$	500	$d^{-1}(g(-COD)m^{-3})$
v_2	Integral time O_2 controller	$\tau_{i(O)}$	0.001	d
v_3	Antiwindup constant O_2 controller	$\tau_{t(O)}$	0.0002	d
v_4	Proportional gain N controller	$K_{(N)}$	15000	$m^3d^{-1}(gNm^{-3})^{-1}$
v_5	Integral time N controller	$\tau_{i(N)}$	0.05	d
v_6	Antiwindup constant N controller	$\tau_{t(N)}$	0.03	d
v_7	Aeration factor ASU1	KLa_1	0	d^{-1}
v_8	Aeration factor ASU2	KLa_2	0	d^{-1}
v_9	Aeration factor ASU3	KLa_3	240	d^{-1}
v_{10}	Aeration factor ASU4	KLa_4	240	d^{-1}
v_{11}	External recycle flow rate	Q_r	18446	m^3d^{-1}
v_{12}	Purge flow rate	Q_w	385	m^3d^{-1}
v_{13}^a	Settler input layer	L_{feed}	5	-

[a]Integer variable.

Table 8.5: Operational variables for the WWTP COST benchmark

For the optimization of this model, convergence curves will be plotted with respect of the number of simulations instead of the computation time. The reason is that, for some simulations, the numerical integration fails, producing an algebraic loop which can involve several hours of computation time. Besides, the overhead introduced by every optimization method can be considered negligible compared to the time needed for each simulation. Due to the small budget of simulations fixed for the problems involving the COST benchmark, the local search was deactivated in our algorithm.

8.3.2 Subproblem WWTP-COST1: PI Tuning

In a first approach, we will try to optimize the control performance of the plant, tested by using the *ISE* (Integral Square Error). Both the nitrate level and oxygen level controllers will be optimized with respect to their controller parameters, that is, the gain K (i.e., v_1 and v_4) and integral time constant τ_i (i.e., v_2 and v_5). The problem is formulated as follows:

$$\min \quad J(v) = c \cdot W^T \cdot ISE \tag{8.44}$$

subject to the system dynamics. $W^T \in \mathbb{R}^{1\times 2}$ contains the weighting coefficients and $ISE \in \mathbb{R}^{2\times 1}$ contains the integral squared errors of the two PI controllers. The weighting vector W^T, the integral square error, *ISE*, and the decision variables vector are as follows:

$$W^T = [w_1 \quad w_2] = \left[\frac{1000}{1001} \quad \frac{1}{1001}\right] \tag{8.45}$$

$$ISE = \begin{bmatrix} ISE_O \\ ISE_N \end{bmatrix} \tag{8.46}$$

$$ISE_{(.)} = \int_{t_0}^{t_f} \varepsilon\,(\tau)^2_{(.)}\, d\tau \tag{8.47}$$

$$v = \begin{bmatrix} v_O \\ v_N \end{bmatrix} = \begin{bmatrix} K_{(O)} \\ \tau_{i(O)} \\ K_{(N)} \\ \tau_{i(N)} \end{bmatrix} \tag{8.48}$$

The weighting vector is chosen such that the ISE_O equals to the ISE_N part when using the benchmark default settings provided by the COST project (Copp, 2001). Boundaries on the decision variables (v^L and v^U) are chosen such that the process dynamics do not show (exceptional) unstable behavior:

$$v^L = [100 \quad 7.0 \cdot 10^{-4} \quad 100 \quad 1.0 \cdot 10^{-2}]^T \tag{8.49}$$

$$v^U = [1000 \quad 7.0 \cdot 10^{-1} \quad 50000 \quad 1.0]^T \tag{8.50}$$

The objective function values are normalized with respect to the performance obtained with the tuned controller settings provided by the COST project (i.e., default values of Table 8.5) using the constant parameter $c = 1.1845 \cdot 10^3$ to obtain a function value equal to one when using default values for the decision variables (i.e., $J(v^{COST} = 1)$).

The histogram (in log-scale) depicting the multistart procedure to check the non-convexity of the problem is shown in Figure 8.5. Due to the high computational cost of every simulation, the number of initial points used in the multistart procedure was only 40. The histogram shows the practical non-convexity of the problem and the best value reported ($f(x) = 0.7463$) outperforms the value obtained with default parameters but not the solutions obtained applying global optimization methods, as shown below.

Table 8.6 shows the results obtained by the different GO methods applied and Figure 8.6 presents the convergence curves for the runs achieving the best solution obtained by every method. The initial point used for this problem is reported together with the best found vectors in Table 8.7. For the sake of comparison, the initial set of 42 points (including the initial one) to create the first surrogate surface was used for *rbfSolve*, *ego* and *SSKm*.

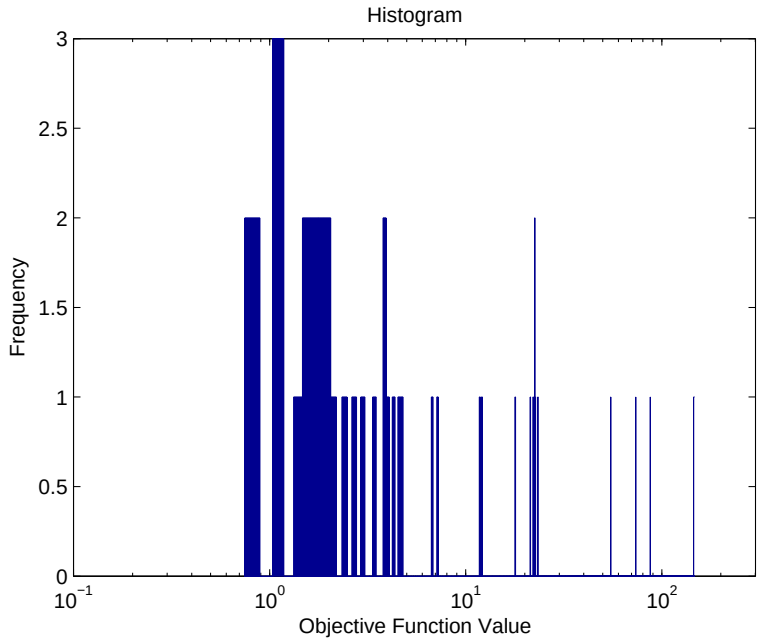


Figure 8.5: Histogram of solutions obtained from the multistart procedure using *fmincon* for problem WWTP-COST1

Solver	Best	Mean	Worst	Mean Evaluations	Mean CPU time (h)
CMAES	0.5313	0.7690	1.8751	402	7.75
DE	0.5399	0.5693	0.6834	400	8.03
glcDirect	0.5565	-	-	400	11.04
OQNLP	0.6350	-	-	400	11.00
SRES	0.6815	1.2895	1.6848	400	8.51
SSm	0.5340	0.6171	0.7672	400	9.65
rbfSolve[a]	0.6530	-	-	388	10.84
rbfSolve[b]	0.5287	-	-	283	8.05
ego	0.7797	-	-	355	9.92
SSKm[c]	0.5293	0.5347	0.5606	400	8.55

[a] Using thin plate splines
[b] Using cubic splines
[c] Results for $p = 0$

Table 8.6: Results for problem WWTP-COST1

Parameter	Best *rbfSolve* solution	Best *SSKm* solution	Initial point
$K_{(O)}$	539.82	470.46	750.62
$\tau_{i(O)}$	$7 \cdot 10^{-4}$	$7 \cdot 10^{-4}$	0.50691
$K_{(N)}$	19975	20246	27831
$\tau_{i(N)}$	0.027052	0.026625	0.093233
J	0.5287	0.5293	35.91

Table 8.7: Best solutions provided by *SSKm* and *rbfSolve* for problem WWTP-COST1

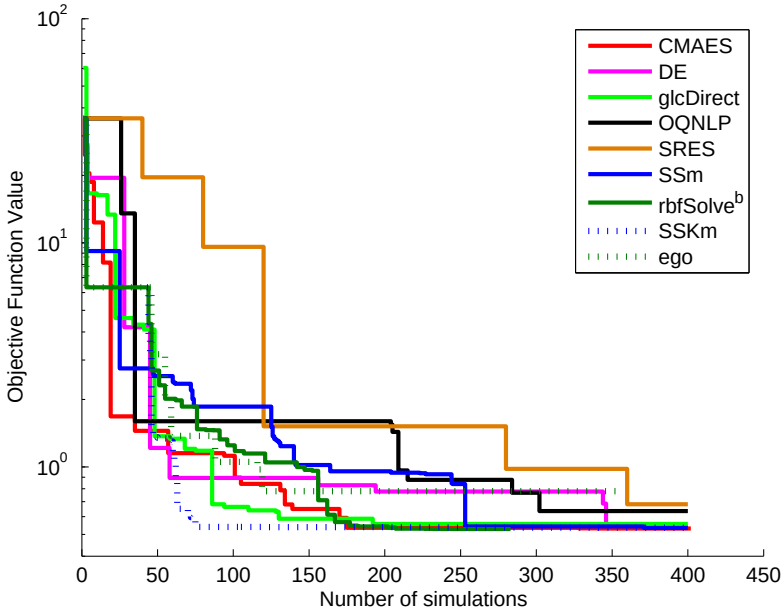


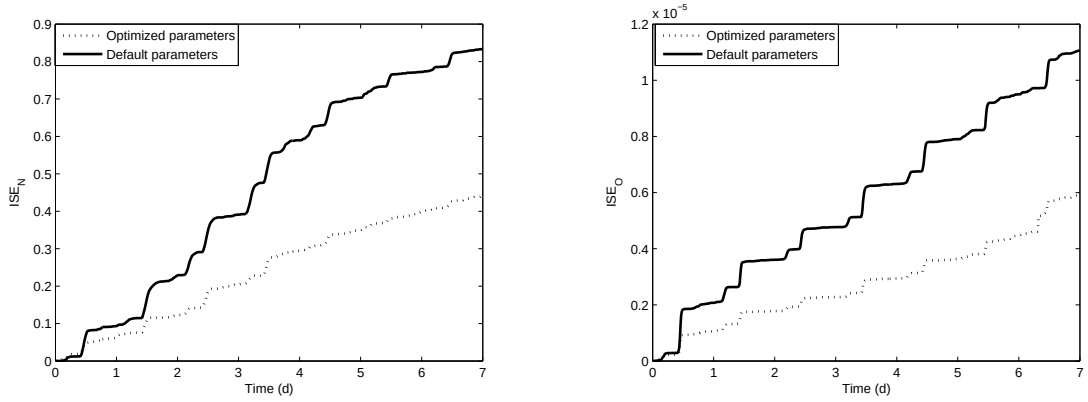
Figure 8.6: Convergence curves for the different solvers in problem WWTP-COST1

As shown in Table 8.6, two of the surrogate model-based solvers (i.e., *rbfSolve* and *SSKm*) present the best results. However, Figure 8.6 shows a faster convergence rate of *SSKm* with respect of the rest of solvers. Indeed, it achieves a solution in the order of the best ones in less than 100 simulations.

To illustrate the improvements achieved by applying global optimization to the minimization of the controllers *ISE*'s, Table 8.8 shows the values obtained with the default parameters provided by the benchmark authors (not with the initial point used for the optimizations) compared with those obtained with the optimized controllers parameters (the best result obtained by *rbfSolve*). The evolution of the *ISE*'s for both solutions is also provided in Figure 8.7.

Index	**Value with** v_{COST}	**Value with** $v_{rbfSolve^b}$
J	0.9985	0.5287
ISE_N	0.8335	0.4415
ISE_0	$1.1055 \cdot 10^{-5}$	$5.9025 \cdot 10^{-6}$

Table 8.8: Initial and optimized indexes for problem WWTP-COST1



(a) N-controller (b) O-controller

Figure 8.7: *ISE* evolution comparison for default and optimized parameters

8.3.3 Operational design

Subproblem WWTP-COST2: NLP Problem

After having tested the different optimization algorithms over the COST benchmark model, the next step is to pose a more complicated problem in terms of design (and also in terms of number of decision variables). The new formulated objective function will be more complex and will take into account not only controllability aspects but also the process economy. The selected decision variables for this extended problem will be, apart from the controllers parameters chosen in the previous section, the aeration factors of the aerated tanks (i.e., KLa_3 and KLa_4), the external recycling flow rate, Q_r, and the sludge purge flow rate, Q_w. The new optimization problem is formulated as follows:

$$\min \quad C(v) = w_1 \cdot \phi_{cont} + w_2 \cdot \phi_{econ} \tag{8.51}$$

where ϕ_{cont} is the same term defined in Equation 8.44. ϕ_{econ} takes into account the different terms which define the operating costs of the process, such as effluent quality, EQ, aeration and pumping energies, AE and PE, and the amount of sludge for disposal, P_{sludge}. Vanrolleghem and Gillot (2002) defined particular economic costs derived from each of these indexes. Based on the relations among these costs, we have defined ϕ_{econ} as:

$$\phi_{econ} = 2 \cdot EQ + AE + PE + 3 \cdot P_{sludge} \tag{8.52}$$

w_1 and w_2 are chosen for both terms, ϕ_{cont} and ϕ_{econ}, to be in the same order of magnitude when using the default values for the decision variables. As in the previous section, the

optimization problem is subject to the system dynamics and the bounds for the decision variables. For the controllers parameters we use the same bounds as in the previous case (i.e., equations 8.49 and 8.50). Upper bounds for these new considered decision variables were chosen taking into account the recommendations of the benchmark authors, whereas the lower bounds were chosen to avoid systematic numerical integration errors along the optimizations. These bounds, together with the initial point used for the optimizations are shown in Table 8.9

Variable	**Lower bound**	**Upper bound**	**Initial Point** ($J = 119430$)
$K_{(O)}$	100	1000	955.12
$\tau_{i(O)}$	0.0007	0.7	0.16234
$K_{(N)}$	100	50000	30381
$\tau_{i(N)}$	0.01	1.0	0.49112
KLa_3	160	360	338.26
KLa_4	160	360	312.42
Q_r	10000	36892	22275
Q_w	100	1844.6	132.28

Table 8.9: Bounds and initial point for WWTP-COST2 problem

The histogram (in log-scale) depicting the multistart procedure to check the multimodality of the problem is shown in Figure 8.8. The number of initial points used was also 40 for the same reasons given in the previous section. The histogram shows the practical multimodality of the problem and the best value reported ($f(x) = 35521$) does not improve the value obtained using default values for the decision variables ($f(x) = 35225$).

Table 8.10 shows the results obtained by the different GO methods applied and Figure 8.9 presents the convergence curves for the best solution obtained by every method. For the sake of comparison, the initial set of 66 points (including the initial one) to create the first surrogate surface was used for *rbfSolve*, *ego* and *SSKm*.

As shown in Table 8.10, only *SSm* and two surrogate model-based methods (i.e., *rbfSolve* and *SSKm*) are able to reduce the function value under 34000. In particular, *SSKm* obtains the best solution for this budget of simulations. A value of $p = 0.5$ seems to be more suitable for this problem, providing a smaller dispersion over the 10 optimizations performed. However, Figure 8.9 shows that, even if *SSKm* and *rbfSolve* provide the best function values, they do not present the fastest convergence rate as expected. The reason might be an inadequate initial sampling for building the first surrogate surface and also the number of initial observations to build it, which are two crucial elements regarding the efficiency of the methods.

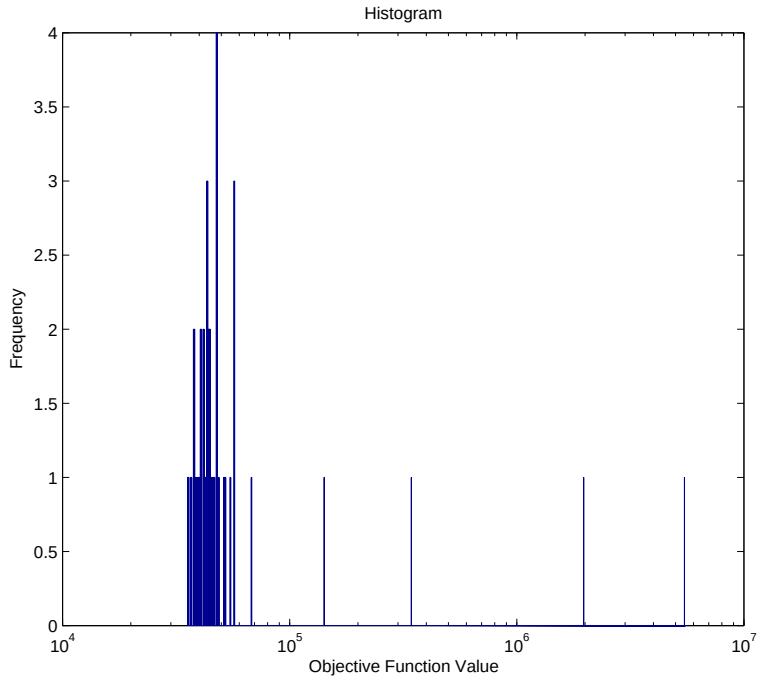


Figure 8.8: Histogram of solutions obtained from the multistart procedure using *fmincon* for WWTP-COST2 problem

Solver	Best	Mean	Worst	Mean Evaluations	Mean CPU time (h)
CMAES	34852	35531	37561	802	24.55
DE	34393	34773	35380	800	27.86
glcDirect	35402	-	-	800	20.72
OQNLP[a]	40902	-	-	800	35.18
SRES	34530	35950	37271	800	27.25
SSm	33926	34690	35574	800	28.04
rbfSolve[b]	34884	-	-	492	39.63
rbfSolve[c]	33970	-	-	677	35.11
ego	34612	-	-	800	24.25
SSKm[d]	33633	41363	36506	800	23.62
SSKm[e]	33544	34223	35383	800	30.05

[a]Local search deactivated
[b]Using thin plate splines
[c]Using cubic splines
[d]Results for $p = 0$
[e]Results for $p = 0.5$

Table 8.10: Results for WWTP-COST2 problem

However, the other surrogate model-based optimization method tested (*ego*) shows a fast initial convergence rate although the final value provided is not as good as those provided by *SSKm* and *rbfSolve*. The best vector found by *SSKm* is the following:

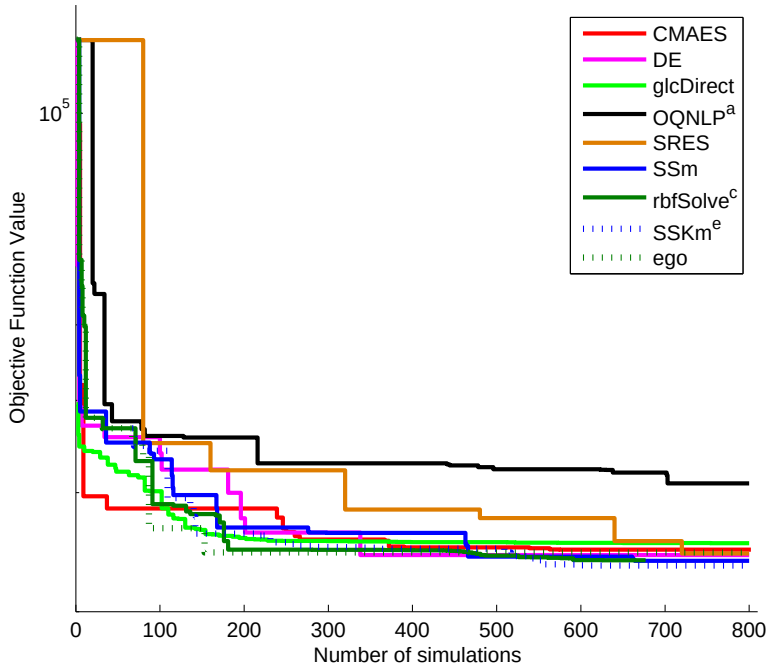


Figure 8.9: Convergence curves for the different solvers in WWTP-COST2 problem

$$v^{SSKm} = [690.046 \quad 0.0009 \quad 19557 \quad 0.02128 \quad 181.88 \quad 184.54 \quad 16624 \quad 370.09]^T \tag{8.53}$$

To illustrate the improvements achieved by applying global optimization to this problem, Table 8.11 shows the values obtained with the default parameters provided by the benchmark authors compared with those obtained with the optimized controllers parameters (the best result obtained by *SSKm*).

Index	**Value with** v_{COST}	**Value with** v_{SSKm^e}
C	35225	33544
ϕ_{cont}	0.9985	0.5888
ISE_N	0.8335	0.4533
ISE_0	$1.1055 \cdot 10^{-5}$	$4.4718 \cdot 10^{-5}$
ϕ_{econ}	34227	32965
EQ (kg poll units/d)	9032	8902
AE (kWh/d)	7173	5577
PE (kWh/d)	1919	2394
P_{sludge} (kg/d)	2347	2397

Table 8.11: Initial and optimized indexes for WWTP-COST2 problem

As it usually occurs in multiobjective optimization problems, some of the objectives can

not be reduced at the same time. Some of the indexes (e.g., PE or P_{sludge}) present worst values after the optimization because they are competing with other indexes to reduce the objective function value. A multiobjective optimization approach for this problem would give us an idea of the different solutions in the pareto front to be able to choose the most interesting amongst them.

Subproblem WWTP-COST3: MINLP problem

To finish this chapter, an extension of the previous problem is proposed. In this case, all the variables shown in Table 8.5 will be used as decision variables. Some of the bounds are extended to allow a possible change of plant configuration (e.g., the anoxic tanks could become aerated and vice versa, changing from a pre-denitrification to a post-denitrification configuration). The objective function used will be the same as in the previous problem. For the sake of comparison with a previous work (Exler et al., 2008), the initial point used for this case will be the default one provided by the benchmark authors, and shown in Table 8.5.

The histogram (in log-scale) depicting the multistart procedure to check the non-convexity of the problem is shown in Figure 8.8. The number of initial points used was also 40 and the local solver used for this case was *misqp*.

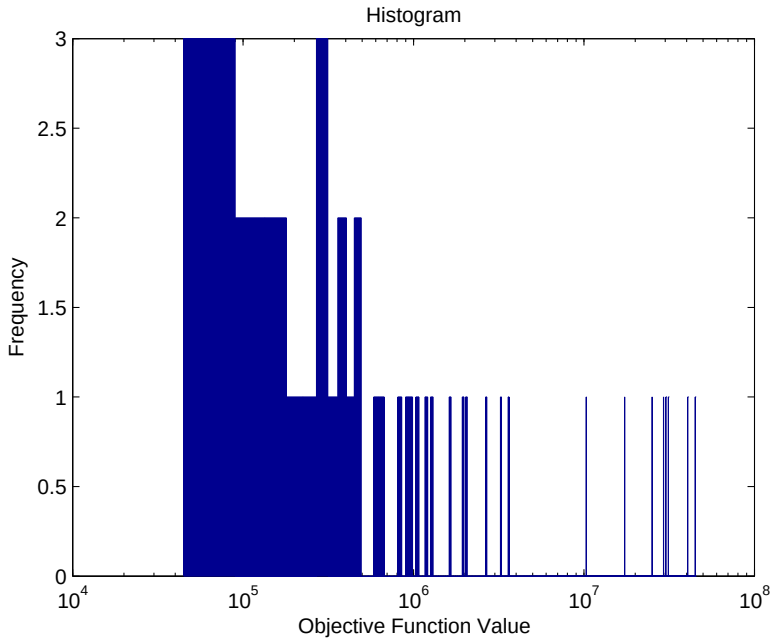


Figure 8.10: Histogram of solutions obtained from the multistart procedure using *misqp* for WWTP-COST3 problem

The histogram shows the practical non-convexity of the problem and the best value re-

ported ($f(x) = 44937$) is very far from the value obtained using default values for the decision variables ($f(x) = 35225$).

The optimization solvers used in this work do not handle integer variables, thus for testing the performance of *SSm*, results obtained by Exler et al. (2008) will be compared with ours. These authors presented results for this problem using a tabu search-based algorithm, *MITS*, and compared their results with those obtained by *OQNLP* and *minlpBB* (Leyffer, 2001). The best *SSm* solution, which provided a function value of 33104, as well as the bounds used for this problems are shown in Table 8.12.

Variable	**Lower bound**	**Upper bound**	x^*_{SSm}
$K_{(O)}$	100	1000	522.71
$\tau_{i(O)}$	0.0007	0.7	0.002537
$\tau_{t(O)}$	0.0001	0.7	0.189337
$K_{(N)}$	100	50000	14366
$\tau_{i(N)}$	0.01	1.0	0.045639
$\tau_{t(N)}$	0.0001	0.07	0.033363
KLa_1	0	360	0
KLa_2	0	360	71.07
KLa_3	0	360	126.02
KLa_4	0	360	183.46
Q_r	0	36892	10316
Q_w	0	1844.6	199.9
$Lfeed$	1	10	7

Table 8.12: Bounds and best *SSm* solution for WWTP-COST3 problem

SSm best solution outperforms the *MITS* results of Exler et al. (2008) which, at the same time, outperformed *OQNLP* and *minlpBB*. It is to note that the solution vector provided by *SSm* achieves a better function value by slightly modifying the default plant configuration (see values of Table 8.5 for comparison). Indeed, some aeration is now introduced in tank 2 whereas it is considerably reduced in tanks 3 and 4. Figure 8.11 shows the convergence curves for these three algorithms in the fixed number of simulations.

Like in the previous example, the improvements in the different performance index applying the optimized vector with respect to the results obtained with default values for the decision variables, are shown in Table 8.13. In this case, savings are mainly produced in the aeration and pumping energies.

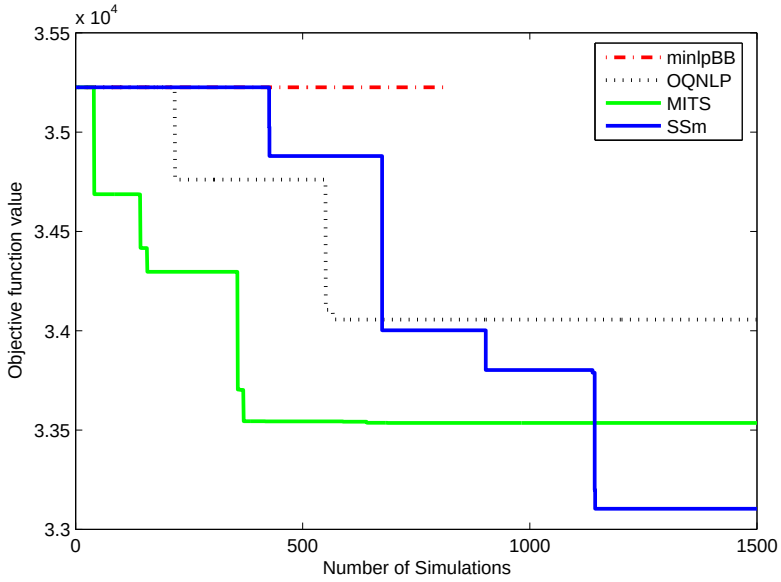


Figure 8.11: Convergence curves for *SSm* compared with those obtained by Exler et al. (2008)

Index	**Value with** v_{COST}	**Value with** v_{SSm}
C	35225	33104
ϕ_{cont}	0.9985	1.2402
ISE_N	0.8335	0.9356
ISE_0	$1.1055 \cdot 10^{-5}$	$1.1334 \cdot 10^{-4}$
ϕ_{econ}	34227	31863
EQ (kg poll units/d)	9032	9185
AE (kWh/d)	7173	4964
PE (kWh/d)	1919	1456
P_{sludge} (kg/d)	2347	2358

Table 8.13: Initial and optimized indexes for WWTP-COST3 problem

8.4 Conclusions

Unlike in the parameter estimation problem section, in which our algorithm is clearly superior to the rest, the differences among solvers are smaller in this set of problems. In any case, our algorithm was competitive, providing the best solution in some cases and showing a low dispersion among its results. In the case of the computationally expensive model (the COST benchmark) the application of *SSKm* led to the best results in the small budget of simulations fixed. It is to note that, for this set of problems, the local search turned out to be inefficient, due to the presence of discontinuities (i.e., in problem WWTP1 because points violating the path constraints are directly rejected using a death-penalty approach as in Moles et al. 2003a) or to the small number of simulations allowed (i.e., in the WWTP-COST benchmark).

Chapter 9

Dynamic optimization problems

9.1 Introduction

Dynamic optimization of bioprocesses has received major attention in recent years. A relevant example is the dynamic optimization of fed-batch bioreactors (Banga et al., 2003a). Dynamic optimization allows the computation of the optimal operating policies to maximize a predefined performance index such as productivity or other economic indexes (Banga et al., 2005). Most bioprocesses present a nonlinear dynamic nature and constraints in both the state and the control variables (see Section 1.1.1), which calls for the use of robust dynamic optimization techniques in order to successfully obtain their optimal operating policies. Numerical methods for the solution of dynamic optimization problems are usually classified under three categories: dynamic programming, indirect and direct approaches. Dynamic programming (Bellman, 2003) is not practically applicable to problems of realistic size or is computationally too expensive. Indirect (classical) approaches are based on the transformation of the original optimal control problem into a two-point boundary value problem (BVP) using the necessary conditions of Pontryagin (Bryson and Ho, 1975). The resulting boundary value problems can be very difficult to solve, especially when state constraints are present. Direct approaches transform the original dynamic optimization problem into a non-linear programming problem using either control vector parameterization (Vassiliadis et al., 1994a,b) or complete parameterization (Cuthrell and Biegler, 1989). From these methods, the control vector parameterization (CVP) approach seems to be the most convenient for dealing with large scale ODE systems (Balsa-Canto et al., 2004), such as those resulting from distributed

systems.

The CVP approach proceeds dividing the time horizon into a number of ρ time intervals. The control variables are then approximated within each interval by means of basic functions, usually low order polynomials (see Figure 9.1), with fixed or variable length over time. This parameterization transforms the original (infinite dimensional) dynamic optimization problem into a non-linear programming problem where the systems dynamics (differential equality constraints) must be integrated for each evaluation of the performance index.

In this work we will use Piecewise Constant approximation, PC (i.e., zero order polynomial) with fixed-length time intervals and fixed final time. Different number of intervals will be used for each problem in order to check the scalability of the different global optimization methods.

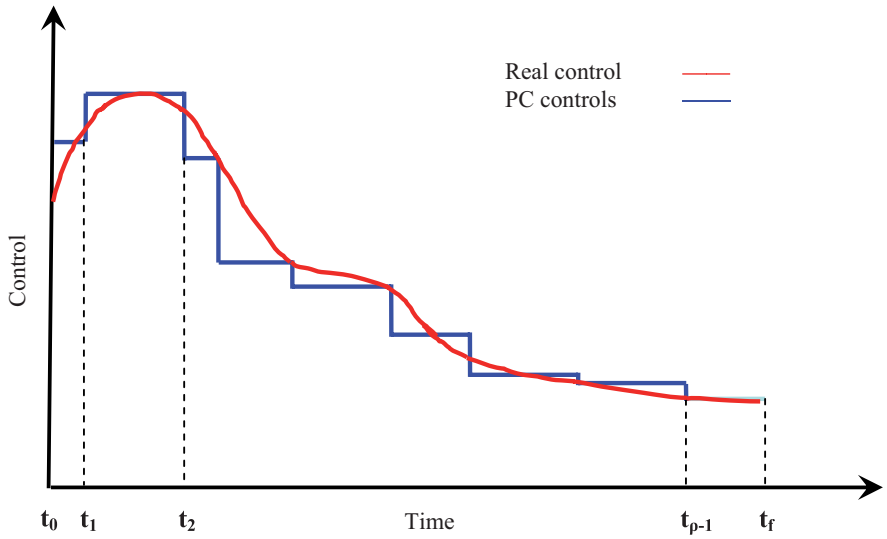


Figure 9.1: Scheme of the CVP approach

NLPs arising from the application of direct approaches (such as CVP) are frequently multimodal. Therefore, gradient based local optimization techniques (such as SQP methods) may converge to local optima.

Global optimization methods are robust alternatives to local methods. Recent advances in global deterministic methods for dynamic optimization have been achieved in recent years (Esposito and Floudas, 2000a; Singer et al., 2001; Papamichail and Adjiman, 2002; Chachuat et al., 2006) but they still need some requirements regarding the functions differentiability and the path constraints type to be handled. Besides, the computational effort is still a barrier for the application of these methods. Stochastic global optimization methods have

been successfully applied to dynamic optimization problems (Banga and Seider, 1996; Banga et al., 1997; Roubos et al., 1999; Rajesh et al., 2001; Sarkar and Modak, 2004; Zhang et al., 2005; Faber et al., 2005; Shelokar et al., 2008). Other approaches using hybrid methods have shown very good results too (Banga and Seider, 1996; Balsa-Canto et al., 2005; Banga et al., 2005).

9.2 Fed-batch reactor for ethanol production

9.2.1 Introduction

This system is a fed-batch bioreactor for the production of ethanol from the anaerobic glucose fermentation by Saccharomyces cerevisiae. The dynamic optimization of this process with fixed final time was studied by Hong (1986), Chen and Hwang (1990a,b) Luus (1993a) and Banga et al. (1997). The objective is to find the feed rate which maximizes the yield of ethanol. Mathematically, it can be stated as:

Find $u(t)$ to maximize

$$J(u) = y_3(t_f)y_4(t_f) \tag{9.1}$$

subject to the system dynamics, described by:

$$\dot{y}_1 = p_1 y_1 - u\left(\frac{y_1}{y_4}\right) \tag{9.2}$$

$$\dot{y}_2 = -10\, p_1 y_1 + u\left(\frac{150 - y_2}{y_4}\right) \tag{9.3}$$

$$\dot{y}_3 = p_2 y_1 - u\left(\frac{y_3}{y_4}\right) \tag{9.4}$$

$$\dot{y}_4 = u \tag{9.5}$$

with

$$p_1 = \left(\frac{0.408}{1 + y_3/16}\right)\left(\frac{y_2}{0.22 + y_2}\right) \tag{9.6}$$

$$p_2 = \left(\frac{1}{1 + y_3/71.5}\right)\left(\frac{y_2}{0.44 + y_2}\right) \tag{9.7}$$

where y_1 represents the microbial population concentration, y_2 is the substrate concentration, y_3 the product concentration (all of them in g/L) and y_4 is the volume (in L), which must satisfy the following end-point constraint: $y_4(t_f) \leq 200$ for $t_f = 54h$ (optimal time calculated

by Chen and Hwang 1990a). The initial state of the system is given by $y(0) = [0 \;\; 150 \;\; 0 \;\; 10]^T$ and the limits for the control variable are: $0 \leq u(t) \leq 12$

9.2.2 Numerical results

For this problem (and the rest of problems considered in this work) several authors proved its non-convex nature. Therefore, the multistart procedure carried out in the previous chapters will not be repeated here. Three levels of discretization were used for every problem in this chapter. In this case we used ρ=10, 20 and 40.

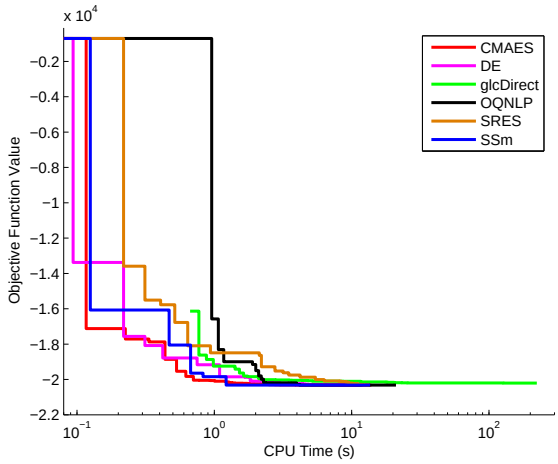
The system of ODE's of this problem was solved using a Runge-Kutta-Felhberg method implemented in the routine *RKF45* (Shampine and Watts, 1977) with absolute and relative integration tolerances of 10^{-7}. From the different local search methods available in *SSm*, SQP-based algorithms were the most competitive for this problem. In particular, *fsqp* and *misqp* provided excellent solutions in a small number of function evaluations. However, *fsqp*'s convergence rate was rather low for the level of discretization $\rho = 40$, therefore *misqp* was the chosen method to perform the local search in this problem. Table 9.1 presents results for every solver with the different levels of discretization.

		CMAES	DE	glcDirect	OQNLP	SRES	SSm
$\rho = 10$	**Best**	**20316.11**	**20316.08**	20203.74	**20316.11**	20305.96	**20316.11**
	Mean	19889.67	20100.72	-	-	20093.14	20291.38
	Worst	18996.02	19672.46	-	-	19554.01	20192.48
$\rho = 20$	**Best**	**20412.14**	20404.36	19738.01	**20412.19**	20327.11	**20412.19**
	Mean	20273.76	20383.95	-	-	20237.58	20412.19
	Worst	19953.39	20341.29	-	-	20095.71	20412.19
$\rho = 40$	**Best**	20430.84	20375.32	19544.88	**20444.47**	20214.40	**20444.86**
	Mean	20360.73	20239.27	-	-	19726.07	20444.86
	Worst	20110.08	19902.08	-	-	19466.64	20444.86

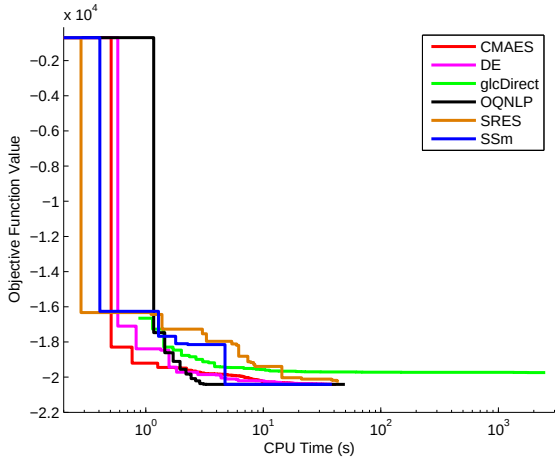
Table 9.1: Results for the ethanol production problem

SSm obtains the best results for this problem for every level of discretization (together with *CMAES* for $\rho = 10$ and *OQNLP* for $\rho = 10$ and 20) with the smallest dispersion in the results. Figure 9.2 shows the convergence curves for all solvers in every discretization level. It must be noted that *CMAES* presents a fast convergence rate compared with the rest of solvers in every case.

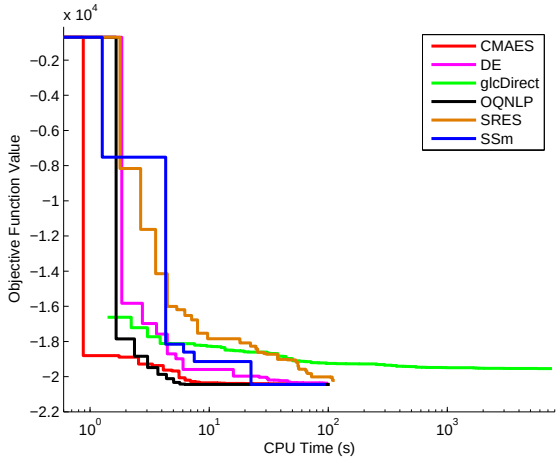
Figure 9.3 shows the control profiles resulting from the best solution found for every discretization level. In this case, all of them correspond to the solutions found by *SSm*.



(a) $\rho = 10$

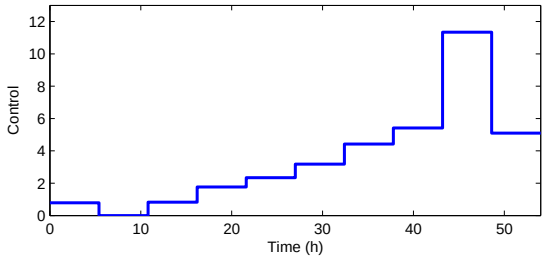


(b) $\rho = 20$

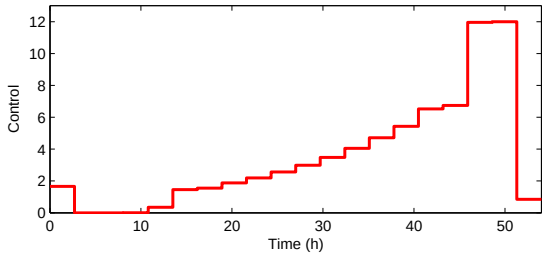


(c) $\rho = 40$

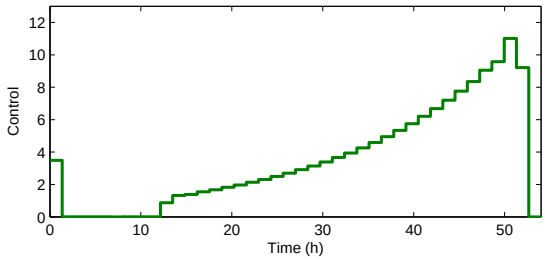
Figure 9.2: Convergence curves for the ethanol production problem



(a) $\rho = 10$, *SSm* solution



(b) $\rho = 20$, *SSm* solution



(c) $\rho = 40$, *SSm* solution

Figure 9.3: Optimal control profiles for the ethanol production problem

9.3 Fed-batch fermenter for penicillin production

9.3.1 Introduction

This problem deals with the dynamic optimization of a fed-batch fermenter for the production of penicillin through anaerobic glucose fermentation. The dynamic optimization of this process with fixed final time was studied by Banga et al. (1997), Cuthrell and Biegler (1989) and Luus (1993b). The optimal control problem is to maximize the total amount of penicillin produced using the feed rate of substrate as the control variable. Mathematically, it can be stated as:

Find $u(t)$ to maximize

$$J(u) = y_2(t_f)y_4(t_f) \tag{9.8}$$

subject to the system dynamics, described by:

$$\dot{y}_1 = h_1 y_1 - \frac{u \cdot y_1}{500 y_4} \tag{9.9}$$

$$\dot{y}_2 = h_2 y_1 - 0.01 y_2 - \frac{u \cdot y_2}{500 y_4} \tag{9.10}$$

$$\dot{y}_3 = -\frac{h_1 y_1}{0.47} - \frac{h_2 y_1}{1.2} - \frac{0.029 y_1 y_3}{0.0001 + y_3} + \frac{u}{y_4}\left(1 - \frac{y_3}{500}\right) \tag{9.11}$$

$$\dot{y}_4 = u/500 \tag{9.12}$$

with

$$h_1 = \frac{0.11 y_3}{0.006 y_1 + y_3} \tag{9.13}$$

$$h_2 = \frac{0.0055 y_3}{0.0001 + y_3(1 + 10 y_3)} \tag{9.14}$$

where y_1, y_2 and y_3 are, respectively, the biomass, penicillin and substrate concentrations (in g/L). y_4 is the fermenter volume (in L). The vector of initial conditions is $y(0) = [1.5 \quad 0 \quad 0 \quad 7]^T$.

The final product is destined to human consumption. Therefore, it must be produced under certain conditions to avoid harmful effects. For that reason, the concentrations of the present species are subject to a set of path constraints, which are:

$$0 \leq y_1 \leq 40 \tag{9.15}$$

$$0 \leq y_3 \leq 25 \tag{9.16}$$

$$0 \leq y_4 \leq 10 \tag{9.17}$$

Bounds for the control variable are defined as $0 \leq u \leq 50$ and the total process time is fixed in $t_f = 132$ h.

9.3.2 Numerical results

The system of ODE's of this problem was solved using *LSODE* (*Livermore solver for ordinary differential equations*, Hindmarsh 1983) with a *BDF* (*Backward Differentiation Formulae*) method, suitable for stiff problems, with absolute and relative integration tolerances of 10^{-7}. For this problem, a direct search local method provided better results than a gradient-based one, thus we used *fminsearch* as local solver. Table 9.2 presents results for every solver with the different levels of discretization.

		CMAES	DE	glcDirect	OQNLP	SRES	SSm
$\rho = 10$	**Best**	**87.934**	**87.934**	87.258	87.775	**87.927**	**87.931**
	Mean	87.837	87.914	-	-	87.688	87.906
	Worst	87.340	87.835	-	-	87.348	87.889
$\rho = 20$	**Best**	87.948	**88.013**	84.490	87.400	87.671	87.998
	Mean	87.841	87.955	-	-	86.900	87.885
	Worst	87.599	87.767	-	-	85.064	87.796
$\rho = 40$	**Best**	87.914	87.926	80.657	87.547	82.709	**87.999**
	Mean	87.861	87.802	-	-	82.709	87.863
	Worst	87.745	87.565	-	-	82.709	87.595

Table 9.2: Results for the penicillin production problem

DE provided the best results and smallest dispersion for the levels of discretization $\rho = 10$ and 20. For $\rho = 40$, *SSm* provided the best solution.

Figure 9.4 shows the convergence curves for all solvers in every discretization level. *OQNLP* presented the fastest initial convergence rate even if the final values provided were not as good as those obtained by other solvers (e.g., *DE*, *SSm* or *CMAES*)

Figure 9.5 shows the control profiles resulting from the best solution found for every discretization level: *DE* for ρ =10, 20 and *SSm* for ρ =40.

9.4 Drying operation

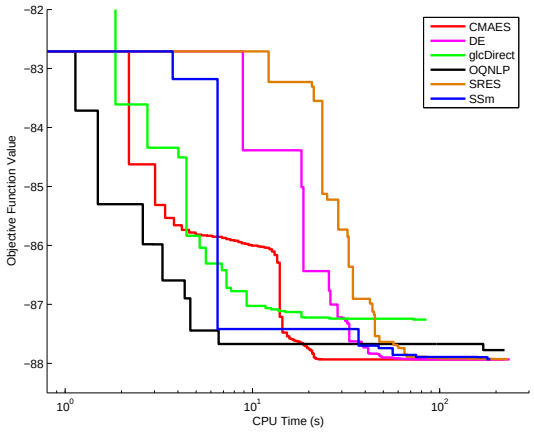
9.4.1 Introduction

In this section we will consider a food convective drying problem, similar to the one formulated by Banga and Singh (1994). In particular, the aim is to dry a cellulose slab (see Figure 9.6) maximizing the retention of a nutrient (ascorbic acid).

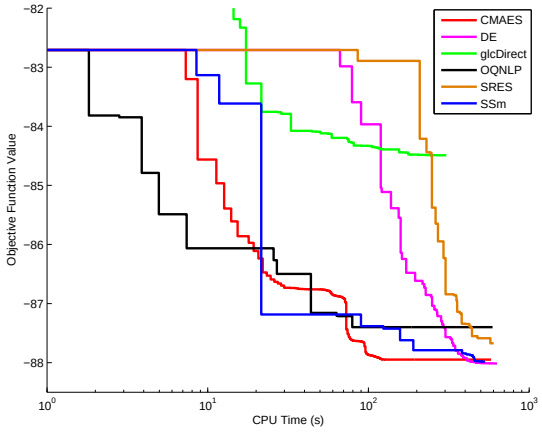
It is assumed that the transport of water within the solid is the controlling mechanism, and that the driving force is the gradient of moisture content. Thus, the governing equation will be Fick's equation for diffusion (Fick's second law):

$$\frac{dm}{dt} = \nabla \left(D \nabla m \right) \tag{9.18}$$

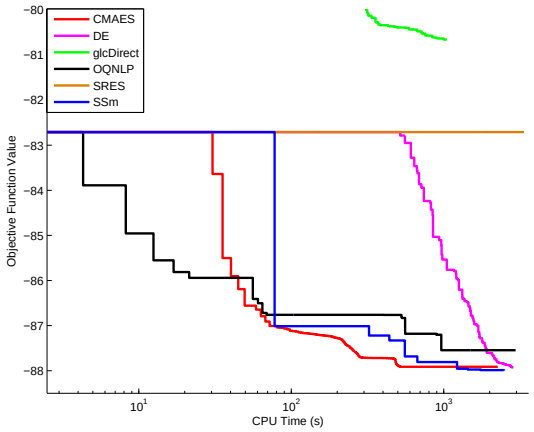
Due to the small thickness of the slab as compared with the other dimensions, we can consider a semi-infinite system where the moisture content depends only on the position with respect to the minor (thickness) dimension. Moreover, in order to take the shrinking effect into account, the diffusivity D is assumed to be a nonlinear function of both the moisture content and the temperature, thus Equation 9.18 reads:



(a) $\rho = 10$

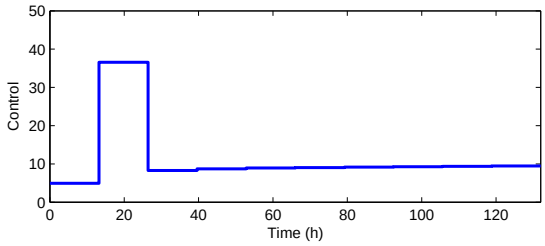


(b) $\rho = 20$

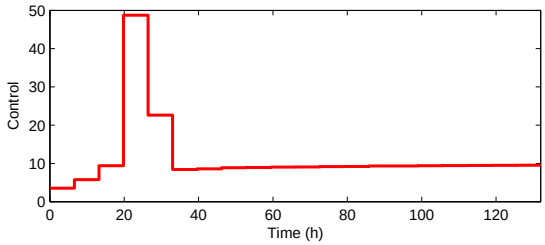


(c) $\rho = 40$

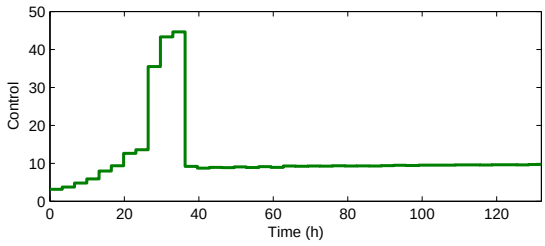
Figure 9.4: Convergence curves for the penicillin production problem



(a) $\rho = 10$, DE solution



(b) $\rho = 20$, DE solution



(c) $\rho = 40$, SSm solution

Figure 9.5: Optimal control profiles for the penicillin production problem

$$\frac{dm}{dt} = D\left(\frac{\partial^2 m}{\partial x^2}\right) + \frac{\partial D}{\partial m}\left(\frac{\partial m}{\partial x}\right)^2 \tag{9.19}$$

being m the moisture content. D is considered to be dependent of the temperature and m. Its value is calculated following Luyben et al. (1982):

$$D = D_{ref} \cdot exp\left[-\frac{E_D}{R}\left(\frac{1}{T_s} - \frac{1}{T_{ref}}\right)\right] \tag{9.20}$$

where D_{ref} y E_D are functions of the moisture content:

$$D_{ref} = exp\left(-\frac{b_1 + b_2 m}{1 + b_3 m}\right) \tag{9.21}$$

$$E_D = \left(\frac{b_4 + b_5 m}{1 + b_6 m}\right) \tag{9.22}$$

The average moisture content of the slab is calculated using:

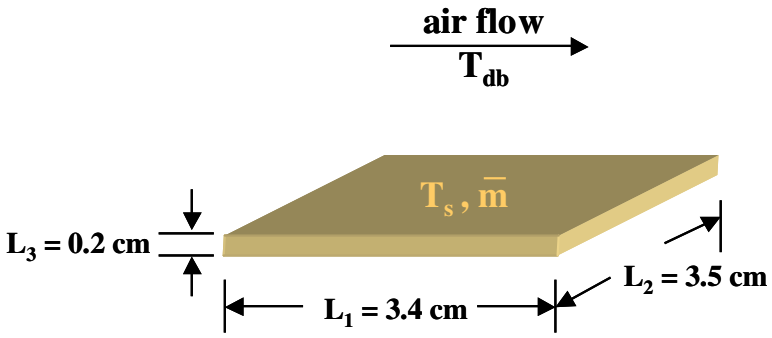


Figure 9.6: Air drying of a cellulose slab

$$m_{avg} = \frac{1}{L}\int_0^L m(x)dx \tag{9.23}$$

$$m_s = \frac{\rho_0 L_1 L_2 L_3}{m_{avg,0} + 1} \tag{9.24}$$

The temperature of the slab is assumed to be uniform. Therefore, an energy balance gives (thin slab assumption):

$$\left(m_s Cp_s + m_s m_{avg} Cp_w\right)\frac{dT_s}{dt} = hA\left(T_{db} - T_s\right) + m_s\lambda_w\left(\frac{dm_{avg}}{dt}\right) \tag{9.25}$$

where the latent heat of vaporization, λ_w, depends on temperature:

$$\lambda_w = \alpha_1 - \alpha_2 T_s \tag{9.26}$$

The heat transfer coefficient and the surface area are variable during drying, so hA is estimated using an empirical linear function of moisture:

$$hA = A_0\left(p_1 m_{avg} + p_2\right) \tag{9.27}$$

where:

$$A_0 = 2\left(L_1L_2 + L_1L_3 + L_2L_3\right) \tag{9.28}$$

where L_1, L_2 and L_3 are the slab dimensions and p_1, p_2 are the model parameters calculated by Mishkin et al. (1982). The nutrient degradation (ascorbic acid) is supposed to be described by first order kinetics (Villota and Karel, 1980a,b).

$$\frac{dC_{AA}}{dt} = -k_{AA}C_{AA} \tag{9.29}$$

where k_{AA} is a function of the temperature and the moisture content:

$$\ln k_{AA} = a_1 m + a_2 T^{-3} + a_3 m^3 + a_4 m^2 T^{-1} + a_5 m T^{-2} + a_6 m^3 T^{-3} + a_7 \tag{9.30}$$

The complete mathematical model is as follows:

$$\Gamma(m, T, m_{avg}, C_{AA}, T_{db}, x, t) = 0 \tag{9.31}$$

The dynamic optimization problem associated with the process consists of finding the dry bulb temperature, T_{db}, along the time to maximize the ascorbic acid retention, ret_{AA}, at the final time, t_f (with t_f=1250 minutes).

$$\max_{T_{db}} ret_{AA}\,(t_f) \tag{9.32}$$

subject to the system dynamics:

$$ret_{AA}\,(t) = \frac{C_{AA,avg}(t)}{C_{AA,0}} = \frac{1}{L_1}\frac{\int_0^{L_1} C_{AA}\,(x,t)\;dx}{C_{AA,0}} \tag{9.33}$$

The problem has an end-point constraint related with the average moisture content at the final time:

$$m_{avg}(t_f) \leq m_{avg,f} \tag{9.34}$$

where $m_{avg,f} = 0.1$ Kg/Kg of dry solid. The bounds for the control variable, T_{db}, are $60 \leq T_{db}(t) \leq 95$ (in ℃).

9.4.2 Numerical results

To solve the system of PDE's describing this model, the numerical method of lines (*NMOL*, Schiesser 1991) was used. The resulting ODE system was solved using *LSODES* (Hindmarsh, 1980) which uses disperse algebra. The integration tolerances (absolute and relative) were

10^{-7}. Due to the inefficiency of the local solvers for this problem, the local search was deactivated for *SSm*, performing a final local refinement with the direct search solver *fminsearch*. *OQNLP* also provided better solutions deactivating its local search, thus only these results are presented in Table 9.3.

		CMAES	DE	glcDirect	OQNLP	SRES	SSm
$\rho = 10$	**Best**	**0.20002**	**0.20003**	0.19979	0.19875	0.20001	**0.20003**
	Mean	0.19710	0.19683	-	-	0.19894	0.19694
	Worst	0.19108	0.18939	-	-	0.19579	0.18742
$\rho = 20$	**Best**	0.19997	0.19913	0.19329	0.15483	0.19989	**0.20010**
	Mean	0.19696	0.19608	-	-	0.19878	0.19687
	Worst	0.19298	0.19185	-	-	0.19728	0.19326
$\rho = 40$	**Best**	**0.19952**	0.19859	0.18848	0.15102	0.19001	0.19788
	Mean	0.19751	0.19442	-	-	0.18796	0.19618
	Worst	0.19522	0.19103	-	-	0.18623	0.19311

Table 9.3: Results for the drying process problem

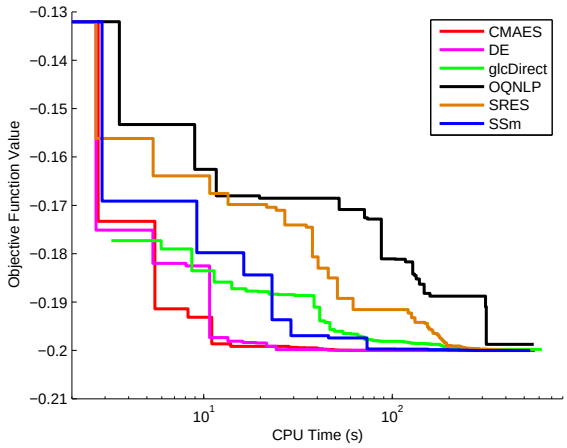
SSm provided the best results for the levels of discretization $\rho = 10$ (with *DE*) and 20. For $\rho = 40$, *CMAES* provided the best solution. It is to note that *SRES* showed the best mean value along the number of runs performed for $\rho = 10$ and 20. Figure 9.7 shows the convergence curves for all solvers in every discretization level.

Figure 9.8 shows the control profiles resulting from the best solution found for every discretization level: *SSm* for $\rho = 10$, 20 and *CMAES* for $\rho = 40$. The control profiles show that smaller levels of discretization (e.g., $\rho = 3$) could achieve similar results in less computational effort. Initial low temperatures avoid a quick degradation of the ascorbic acid and favor the internal heat diffusion and water transport inside the food. The moisture content raises during this first stage and, therefore, the temperature needs to be increased to obtain the average moisture content at the final time imposed by the constraint.

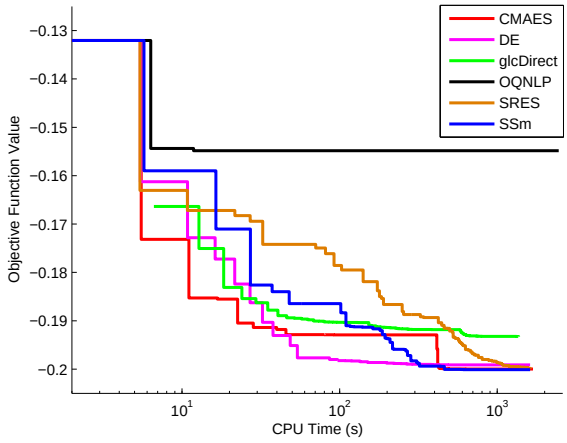
9.5 Microwave heating of foods

9.5.1 Introduction

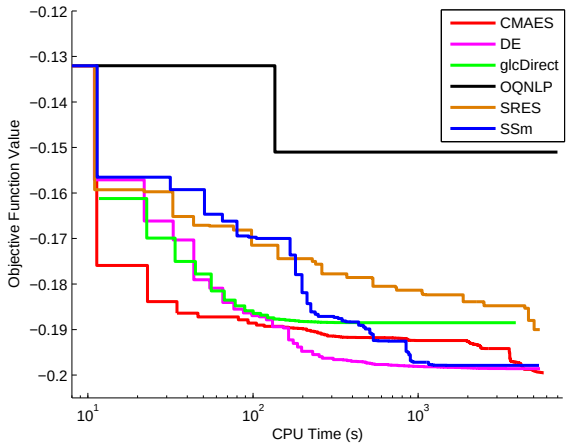
This section deals with the optimization of the heating process of foods in a combined microwave-convection oven. In particular, the product considered is a cylinder with radius R_m and height Z_m. These heating mechanisms complement each other: while microwaves favor the internal heating, convection acts in the surface. Thus, both mechanisms may be optimally combined to maximize the uniformity in the product temperature, avoiding cold or hot points (typical in microwave processing) and their possible effects over the security



(a) $\rho = 10$

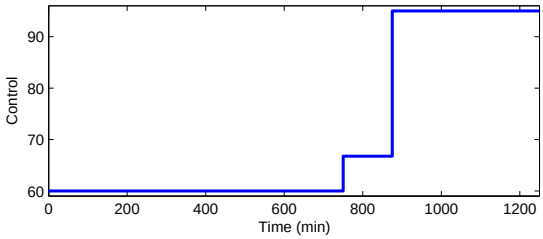


(b) $\rho = 20$

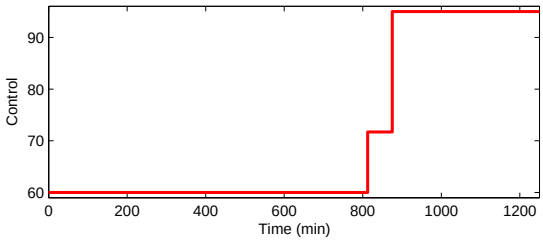


(c) $\rho = 40$

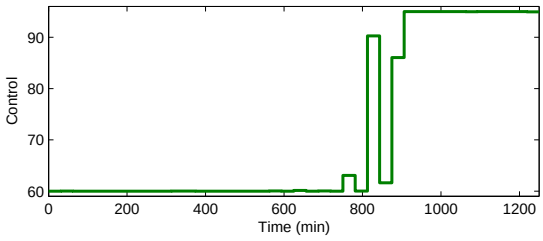
Figure 9.7: Convergence curves for the drying process problem



(a) $\rho = 10$, *SSm* solution



(b) $\rho = 20$, *SSm* solution



(c) $\rho = 40$, *CMAES* solution

Figure 9.8: Control profiles for the drying process problem

and/or quality of the final product.

We will try to solve an optimal control problem similar to the one proposed by Saa et al. (1998) and Banga et al. (1999), where the objective is to obtain the profiles for the microwave power, $P_0(t)$, and the oven temperature, $T_{oven}(t)$ to maximize the uniformity of the final temperature, for a final time of 270 seconds.

In particular, the mathematical model used was formulated by Lin et al. (1995) in which it is assumed that the equation governing the heating process in a combined oven is the second Fourier's law with a term, Φ, of energy generation arising from the microwave heating:

$$C_p\tilde{\rho}\frac{\partial T}{\partial t} = k\left(\frac{1}{r}\frac{\partial T}{\partial r} + \frac{\partial^2 T}{\partial r^2} + \frac{\partial^2 T}{\partial z^2}\right) + \Phi \tag{9.35}$$

The thermal conductivity, k, is assumed to be constant and the generation term, Φ, is based on Lambert's law (Ohlsson and Bengtsson, 1971). Besides, Lin et al. (1995) added a correction term due to the internal wave reflection. Here it is assumed that the product has

a uniform initial temperature, $T(r,z) = T_0 = 25$ ℃. Heating by convection is imposed by the boundary conditions

$$k\frac{\partial T}{\partial r} = h\left(T_{oven} - T\right) \quad \text{for} \quad r = R_m \tag{9.36}$$

$$k\frac{\partial T}{\partial z} = h\left(T_{oven} - T\right) \quad \text{for} \quad z = Z_m \tag{9.37}$$

Due the radial and axial symmetry, the internal temperature distribution of the product can be fully described with the temperature change in a quarter of the transverse section of the z axis. Besides, the product is considered homogeneous, as expressed by the following equations:

$$\frac{\partial T}{\partial r} = 0 \quad \text{for} \quad r = 0 \tag{9.38}$$

$$\frac{\partial T}{\partial z} = 0 \quad \text{for} \quad z = 0 \tag{9.39}$$

The incident power in the product's surface is considered as constant and orthogonal to it. The term of heat generation for a cylindrical geometry is given by the equations

$$\Phi_r(r,T) = 2\beta(T)\frac{R_m}{r}P_r\left[e^{-2\beta(T)(R_m-r)} + e^{-2\beta(T)(R_m+r)}\right] \tag{9.40}$$

$$\Phi_z(z,T) = 2\beta(T)P_z\left[e^{-2\beta(T)(Z_m-z)} + e^{-2\beta(T)(Z_m+z)}\right] \tag{9.41}$$

$$\Phi(r,z,T) = \Phi_r(r,T) + \Phi_z(z,T) \tag{9.42}$$

with

$$P_r = P_0/\left(4\pi R_m Z_m\right) \tag{9.43}$$

$$P_z = 0.107P_0/\left(8\pi R_m R_m\right) \tag{9.44}$$

To avoid the central singularity of Equation 9.40 when $r \to 0$, the radial component of the heat generation term is calculated with

$$\Phi_r(r,T) = \begin{cases} 2\beta(T)\frac{R_m}{r}P_r\left[e^{-2\beta(T)(R_m-r)} + e^{-2\beta(T)(R_m+r)}\right] & \text{if} \quad r > \varepsilon \\ 2\beta(T)\frac{R_m}{\varepsilon}P_r\left[e^{-2\beta(T)(R_m-\varepsilon)} + e^{-2\beta(T)(R_m+\varepsilon)}\right] & \text{if} \quad r \leq \varepsilon \end{cases} \tag{9.45}$$

with $\varepsilon = 3.5 \cdot 10^{-2} \cdot R_m$. Themophysical parameters and product dimensions were taken from Chen et al. (1993). The mathematical model described above can be formulated as:

$$\Psi(T, P_0, T_{oven}, r, z, t) = 0 \tag{9.46}$$

The optimal control problem is formulated to find $P_0(t)$ and $T_{oven}(t)$ to minimize:

$$J = T_{dif}(t_f) = T_{\max}(t_f) - T_{\min}(t_f) \tag{9.47}$$

subject to the system dynamic and an inequality constraint related with the minimum desired final temperature:

$$T_{\min}(t_f) \geq T_{\min}^{SET} = 60 \tag{9.48}$$

The bounds for the control variables are $0 \leq P_0 \leq 190$ (in W) and $25 \leq T_{oven} \leq 200$ (in ℃).

9.5.2 Numerical results

Like in the previous example, the system of PDE's describing this model was transformed into an ODE's system using the numerical method of lines (*NMOL*) and was solved using LSODES with integration tolerances (absolute and relative) of 10^{-7}. Also, the local search was deactivated for *SSm*, performing a final local refinement with the direct search solver *fminsearch*. Since there are two control variables in this example, the discretization levels considered are $\rho = 5$, 10 and 20, in order to evaluate the same number of decision variables used in the previous case studies of this chapter. Table 9.4 shows the results obtained by every solver for the different levels of discretization.

		CMAES	DE	glcDirect	OQNLP	SRES	SSm
$\rho = 5$	**Best**	3.6691	**3.6665**	6.5338	3.6988	3.6702	3.6718
	Mean	3.6780	3.6685	-	-	3.6992	3.6816
	Worst	3.6938	3.6793	-	-	3.7829	3.7268
$\rho = 10$	**Best**	3.4526	**3.4351**	6.6787	3.4784	3.4578	3.4374
	Mean	3.4684	3.4422	-	-	3.4733	3.4498
	Worst	3.4963	3.4537	-	-	3.5073	3.4626
$\rho = 20$	**Best**	3.3840	3.3680	6.6859	3.4921	3.9809	**3.3615**
	Mean	3.4891	3.3886	-	-	4.0575	3.4522
	Worst	3.6759	3.4095	-	-	4.1485	3.9080

Table 9.4: Results for the combined oven problem

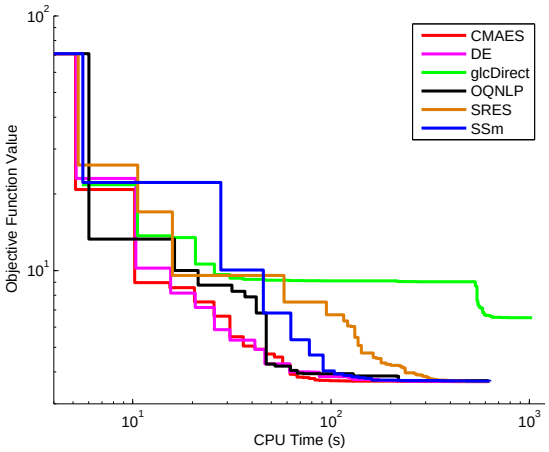
DE provided the best results for the levels of discretization $\rho = 5$ and 10, and the best mean value for all cases. For $\rho = 20$, *SSm* provided the best solution. Figure 9.9 shows the

convergence curves for all solvers in every discretization level.

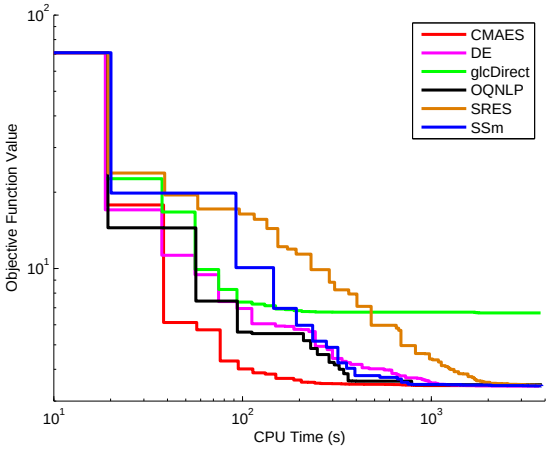
Figure 9.10 shows the control profiles resulting from the best solution found for every discretization level: *DE* for $\rho = 5$, 10 and *SSm* for $\rho = 20$. The optimal control profiles for the microwave power show the same behavior in every case and are easy to implement in practice.

9.6 Conclusions

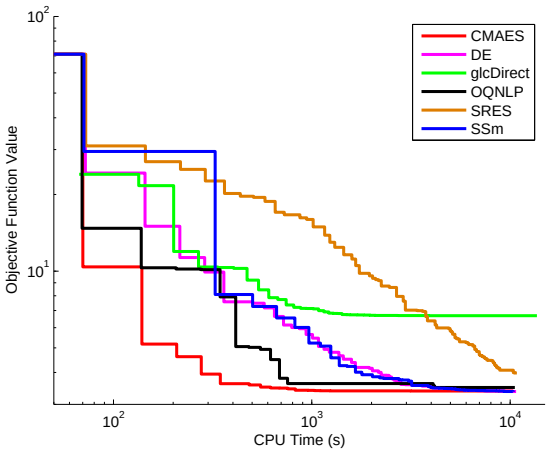
The results obtained in this section show that our algorithm is also competitive for dynamic optimization problems, achieving the best results in most of the cases and showing the best mean values in many of them. Providing the different discretization levels considered, it has also proved to have a good scalability with the problem size. The algorithm *DE* performs very well in this set of problems too, regarding both best and mean values. According to the convergence curves, *CMAES* presents the fastest convergence rate in general, even if the final results are not as good as those provided by *SSm* and *DE*.



(a) $\rho = 5$

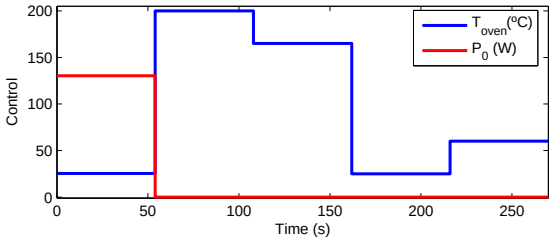


(b) $\rho = 10$

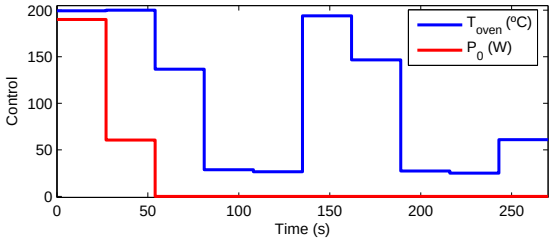


(c) $\rho = 20$

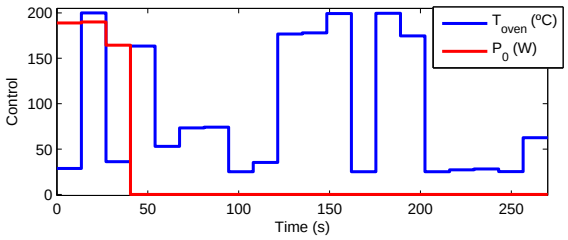
Figure 9.9: Convergence curves for the combined oven problem



(a) $\rho = 5$, *DE* solution



(b) $\rho = 10$, *DE* solution



(c) $\rho = 20$, *SSm* solution

Figure 9.10: Control profiles for the combined oven problem

Chapter 10

Executive summary of results

In this chapter we provide a summary of the results obtained in Chapters 7, 8 and 9. The table below shows a summary of the performance of the different algorithms used in this work in the different types of problems tested, reporting the number of times that each solver obtained the best solution and the best mean value along the different runs (only for continuous problems). *SSm* obtains the highest score for both best and mean values. Besides, for the MINLP problem arising from the operational design of the WWT COST benchmark model, *SSm* was able to outperform the best published solutions.

A more rigorous comparison can be done by making use of the performance profiles methodology (Dolan and Moré, 2002). Following Auger and Hansen (2005), we define the success performance FE for a solver on a specific problem by:

$$\mathsf{FE} = eval_{mean} \cdot \frac{\#\text{all runs}(10)}{\#\text{successful runs}} \tag{10.1}$$

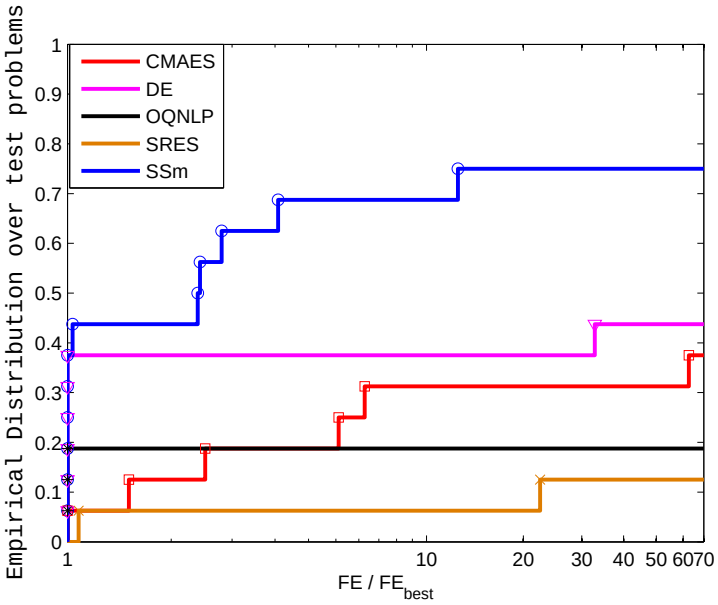
where a run is considered successful if it obtained the optimal solution with a relative error $\leq 0.1\%$ (in our problems, we consider it as the best solution found by any of the solvers). With this definition the best success performance FEbest is given by the lowest value of FE for every problem. The figure below shows the empirical distribution function of the success performance FE/FEbest over all the problems.

Performance profiles methodology ranks *SSm* in the first place for the set of problems considered in the third part of this work compared with the rest of solvers tested.

Solver	Type of problems	# best	# mean
CMAES	Parameter estimation	0	0
	Integrated design and control	1	0
	Dynamic optimization	5	1
	Total	**6**	**1**
DE	Parameter estimation	0	0
	Integrated design and control	2	0
	Dynamic optimization	5	5
	Total	**7**	**5**
glcDirect	Parameter estimation	0	-
	Integrated design and control	0	-
	Dynamic optimization	0	-
	Total	**0**	**-**
OQNLP	Parameter estimation	0	-
	Integrated design and control	0	-
	Dynamic optimization	2	-
	Total	**2**	**-**
SRES	Parameter estimation	0	0
	Integrated design and control	1	0
	Dynamic optimization	0	2
	Total	**1**	**2**
SSm	Parameter estimation	3	3
	Integrated design and control	1	0
	Dynamic optimization	7	4
	Total	**11**	**7**
rbfSolve[a]	Integrated design and control	1	-
ego[a]	Integrated design and control	0	-
SSKm[a]	Integrated design and control	1	2

[a]Only applied for the WWT COST benchmark.

Summary of results



Performance profiles

Part IV

Conclusions

Conclusions

This work deals with the global optimization of processes related with biotechnological and food industries. Due to the structure of the mathematical models describing these processes, the optimization of these systems is a complex task.

Here we have developed a scatter search-based methodology for mixed-integer nonlinear optimization problems, which intends to be effective for solving global optimization problems from the biotechnological and food industries. The procedure treats the objective function as a black box, making the search algorithm context-independent. We have expanded and advanced knowledge associated with the implementation of scatter search procedures. In particular:

- We have used a more general solution combination method which drives the search to other directions of the search space apart from those defined by every pair of solutions in the population.

- Different filters to prevent premature stagnation and/or getting stuck in flat areas have been included.

- A new population rebuilding strategy taking into account relative search directions has also been developed.

- We have designed different mechanisms to intensify the search such as the as the "intensification" and the *go beyond* strategy.

- The procedure has also been extended to handle integer variables as well as continuous ones.

- The methodology has been implemented in Matlab under the name of *SSm* (*Scatter Search for Matlab*) and has successfully been tested over a set of benchmark problems.

Regarding the global optimization of computationally expensive process models:

- We have combined kriging and scatter search methodologies to develop a global optimization method for computationally expensive process models. The evolutionary framework of the scatter search procedure automatically selects a set of points that balance between intensification and diversification in which the kriging prediction is performed.

- We have also proposed a new performance index making use of the statistical information provided by kriging, and different patterns of search with guidelines depending on the type of optimization problems faced.

- The method, through its associated software tool implemented in Matlab, *SSKm* (*Scatter search with kriging for Matlab*), has been tested over a set of two benchmark mathematical functions and has proved to be efficient in locating not only the global optimum but all of them (if they exist) in a few number of function evaluations.

In the last part of this work, the proposed methodologies described above have been tested by applying their software implementations to different global optimization problems from the biotechnological and food industries, covering the most relevant type of problems arising in these areas (i.e., parameter estimation, integrated design and control and dynamic optimization). In order to have an idea about their efficiency, they have been compared with other state-of-the-art global optimization methods. The results obtained have led to the following conclusions:

- The proposed methodologies are efficient and robust for the kind of problems intended to solve, specially in the case of parameter estimation problems.

- For some problems (i.e., the estimation of parameters in a biochemical pathway or the integrated design of the WWT COST benchmark), the results obtained outperformed the best solutions found in literature (for the first case, in several orders of magnitude in terms of convergence rate).

- In all cases our algorithms were competitive, being the most efficient among the tested ones in many of the examples.

Future research will be focused on the auto-tuning of the methods parameters, as well as on testing of new designs for some of their parts. For the case of the algorithm for computationally expensive process models, a dynamic population could be useful at the beginning and at the end of the search. The use of a different covariance for big problems should also be considered for problems with a high number of variables (e.g., > 50). A transformation of the data (e.g., log or inverse) could be also added for those cases in which the correlation is not adequate.

Conclusiones

Este trabajo trata sobre la optimización global de procesos relacionados con las industrias biotecnológica y alimentaria. Debido a la estructura de los modelos matemáticos que describen dichos procesos, la optimización de estos sistemas es una tarea compleja.

Se ha desarrollado un método basado en búsqueda dispersa (*scatter search* en inglés) para problemas de optimización no lineal mixta entera, cuyo propósito es ser efectiva para resolver problemas de optimización global de industrias biotecnológicas y alimentarias. El método trata la función objectivo como una caja negra, haciendo al algoritmo de búsqueda independiente del tipo de problema tratado. Se ha ampliado el conocimiento asociado a la implementación de algortimos basados en *scatter search*. En concreto:

- Se ha implementado un método de combinación de soluciones más general que dirige la búsqueda hacia otras direcciones distintas a las definidas por cada par de soluciones en el conjunto de referencia (*RefSet*).
- Se han incluido diferentes filtros prevenir el estancamiento prematuro de las soluciones y para evitar quedar atrapados en zonas planas del espacio de búsqueda.
- Se ha desarrollado una nueva estrategia de regeneración de la población que tiene en cuenta las diferentes direcciones relativas de búsqueda.
- Se han diseñado diferentes mecanismos para intensificar la búsqueda.
- El método se ha extendido para el manejo de variables enteras además de continuas.
- La metodología se ha implemtado en Matlab bajo el nombre de *SSm* (*Scatter Search for Matlab* en inglés) y ha sido testado de forma satisfactoria sobre un conjunto de problemas de banco de pruebas.

Respecto a la optimización global de procesos computacionalmente costosos de simular:

- Se han combinado las metodologías de *scatter search* y *kriging* para desarrollar un algoritmo de optimización global para modelos computacionalmente costosos. El marco evolutivo proporcionado por *scatter search* hace que el método seleccione automáticamente un conjunto de puntos que aportan tanto intensificación como diversificación en la búsqueda, en los cuales se lleva a cabo la predicción por *kriging*.

- Se ha propuesto un nuevo índice de evaluación que hace uso de la información estadística proporcionada por el *kriging*, y diferentes patrones de búsqueda con indicaciones dependiendo del tipo de problema tratado.

- El método, mediante su herramienta de software asociada implementada en Matlab, *SSKm* (*Scatter search with kriging for Matlab* en inglés), ha sido probada sobre varios problemas de banco de pruebas, mostrando su eficiencia para localizar no sólo el óptimo global sino todos los que haya (si los hay) en un reducido número de evaluaciones de la función objetivo.

En la última parte de este trabajo, las metodologías propuestas han sido evaluadas mediante sus implementaciones en software, aplicándolas a diferentes problemas de optimización global de procesos biotecnológicos y alimentarios, cubriendo los tipos de problemas más relevantes que surgen en estas áreas (estimación de parámetros, diseño integrado con control y optimización dinámica). Para tener una idea de la eficiencia de los métodos, estos han sido comparados con otros algoritmos de optimización global que constituyen el estado actual. Los resultados obtenidos han conducido a las siguientes conclusiones:

- Las metodologías propuestas son eficientes y robustas para el tipo de problemas que se quiere resolver, especialmente en el caso de problemas de estimación de parámetros.

- Para algunos problemas (por ejemplo, la estimación de parámetros en una ruta bioquímica o el diseño integrado del modelo COST), los resultados obtenidos superan las mejores soluciones encontradas en la bibliografía (en el primer caso, en dos órdenes de magnitud en términos de velocidad de convergencia).

- En todos los casos los algoritmos propuestos son competitivos, siendo los más eficientes de entre todos los métodos testados en la mayoría de los casos.

El trabajo futuro se centrará en el autoajuste de los parámetros de los métodos propuestos, así como en el testeo de nuevos diseños para algunas de sus partes. Para el caso del algoritmo

para modelos computacionalmente costosos, un tamaño de población variable podría ser útil al principio y al final de la búsqueda. El uso de un modelo de covarianza diferente para problemas de gran tamaño debería ser considerado también para problemas con elevado número de variables (por ejemplo, > 50). Una transformación de los datos (por ejemplo, logarítmica o inversa) podría ser beneficiosa en los casos en los que la correlación no es adecuada.

Part V

Appendices

Appendix A

Software documentation

A.1 Introduction

SSm seeks the global minimum of a MINLP problem specified by

$$\min_x f(x, p_1, p_2, ..., p_n)$$

subject to

$$\begin{aligned} c_{eq} &= 0 \\ c_L \leq c(x) &\leq c_U \\ x_L \leq x &\leq x_U \end{aligned}$$

where x is the vector of decision variables, and x_L and x_U its respective bounds. $p_1, \ldots, p_n$ are optional extra input parameters to be passed to the objective function (see examples in Sections A.4.3 and A.4.5). c_{eq} is a set of equality constraints. $c(x)$ is a set of inequality constraints with lower and upper bounds, c_L and c_U. Finally, $f(x, p_1, p_2, ..., p_n)$ is the objective function to be minimized.

A.2 *SSm* toolbox

A.2.1 *SSm* problem definition

Setting	Description
problem.f	String containing the name of the objective function
problem.x_L	Vector containing the lower bounds of the variables
problem.x_U	Vector containing the upper bounds of the variables
problem.x_0	Vector containing the given initial point (optional)
problem.neq	Number of equality constraints[a]
problem.c_L	Vector defining the lower bounds of the inequality constraints
problem.c_U	Vector defining the upper bounds of the inequality constraints
problem.int_var	Number of integer variables[b]
problem.bin_var	Number of binary variables[b]
problem.vtr	Objective function value to be reached (optional)

[a] In problems with equality constraints they must be declared before inequality constraints
[b] For mixed integer problems, the variables must be defined in the following order: [cont., int., bin.]

Table A.1: *SSm* problem settings

A.2.2 User options

Option	Description	Default
opts.maxeval	Maximum number of function evaluations	1000
opts.maxtime	Maximum CPU time in seconds	60
opts.iterprint	Print each iteration on screen: 0: Deactivated; 1: Activated	1
opts.plot	Plots convergence curves: 0: Deactivated; 1: Real time; 2: Final results	0
opts.weight	Weight that multiplies the penalty term added to the objective function in constrained problems	10^6
opts.log_var	Indexes of the variables which will be analyzed using a logarithmic distribution instead of a uniform one	[ ]
opts.tolc	Maximum violation constraints violation allowed. This is also used as a tolerance for the local search	10^{-5}
opts.save_report	Saves Results, problem and options in a .mat file 0: Does not save report; 1: Saves report	0
opts.report_name	String specifying the report name	'ssm_report.mat'

Table A.2: *SSm* user options

A.2.3 Global options

Option	Description	Default
opts.dim_refset	Number of elements in *RefSet*	*'auto'* $\left(\frac{d^2-d}{10 \cdot nvar} \geq 0\right)$
opts.ndiverse	Number of solutions generation by the diversificator in the initial stage	*'auto'* $(10 \cdot nvar)$

opts.initiate	Type of *RefSet* initialization: 0: Take bounds, middle point and fill by Euclidean distance 1: Evaluate all the diverse solutions, take the $dim\_refset/2$ best solutions and fill by Euclidean distance	0
opts.combination	Type of combination of *RefSet* elements: 1: Hyper-rectangles combinations 2: Linear combinations	1
opts.regenerate	Type of *RefSet* regeneration: 1: Regeneration by distance diversity 2: Regeneration by direction diversity 3: Randomly alternates 1 and 2	3
opts.delete	Number of *RefSet* elements deleted when regenerating the *RefSet* *'standard'*: Delete $dim\_refset/2$ (the worst half *RefSet* members) *'aggressive'*: Delete $dim\_refset - 1$ (all of them except the best solution found)	*'standard'*
opts.intens	Iteration interval between intensifications	10
opts.tolf	Function tolerance for joining the *RefSet*	10^{-4}
opts.diverse_criteria	Criteria for diversification in the *RefSet* 1: Euclidean distance; 2: Tolerances	1
opts.tolx	Variable tolerance for joining the *RefSet* when the Euclidean distance is deactivated	10^{-3} for all variables

Table A.3: *SSm* global options

A.2.4 Local options

Option	Description	Default
opts.local.solver	Solver to perform the local search 0 (No local search); 'fmincon'; 'constrnew';'nomad'; 'solnp' 'n2fb'; 'dn2fb';'dhc'; 'fsqp';'ipopt','misqp','lsqnonlin'	'fmincon'
opts.local.tol	Level of tolerance in local search: 1: Relaxed; 2: Medium; 3: Tight	2 (3 in final stage)
opts.local.iterprint	Print each iteration of local solver on screen (only for local solvers that allow it): 0: Deactivated; 1: Activated	0
opts.local.n1	Number of function evaluations before applying local search for the first time	$100 \cdot nvar$
opts.local.n2	Minimum number of function evaluations in the global phase between two local searches	$200 \cdot nvar$
opts.local.finish	Applies local search to the best solution found once the optimization is finished	same as *opts.local.solver*
opts.local.bestx	Applies only local search to the best solution found to date (if it is not a local minimum), ignoring filters: 0: Deactivated; 1: Activated	0
opts.local.merit_filter	Activation of merit filter for local search: 0: Deactivated; 1: Activated	1
opts.local.distance_filter	Activation of distance filter for local search: 0: Deactivated; 1: Activated	1
opts.local.thfactor	Merit filter relaxation parameter	0.2
opts.local.maxdistfactor	Distance filter relaxation parameter	0.2

opts.local.wait_maxdist_limit	Apply distance filter relaxation after this number of function evaluations without success in passing the filter	20
opts.local.wait_th_limit	Apply merit filter relaxation after this number of function evaluations without success in passing the filter	20

Table A.4: *SSm* local options

When using *n2fb* (or *dn2fb*) and *lsqnonlin* as local solvers, the objective function value must be formulated as the square of the sum of differences between the experimental and predicted data (i.e., $\sum_{i=1}^{ndata}(yexp_i - yteor_i)^2$). Besides, a third output argument must be defined in the objective function: a vector containing those residuals (i.e., $R = [(yexp_1 - yteor_1), (yexp_2 - yteor_2), \ldots, (yexp_{ndata} - yteor_{ndata})]$). In Section A.4.5 an application example illustrates the use of these local methods.

A.2.5 *SSm* output

SSm's output is a structure (called **Results** by default) containing the following fields:

- **Results.fbest**: Best objective function value found.
- **Results.xbest**: Vector providing the best function value found.
- **Results.cpu_time**: CPU Time (in seconds) consumed in the optimization.
- **Results.f**: Vector containing the best objective function value after each iteration.
- **Results.x**: Matrix containing the best vector after each iteration.
- **Results.time**: Vector containing the CPU time consumed after each iteration.
- **Results.neval**: Vector containing the number of evaluations after each iteration.
- **Results.numeval**: Total number of function evaluations.
- **Results.local_solutions**: Matrix of local solutions found.
- **Results.local_solutions_values**: Function values of the local solutions.
- **Results.end_crit**: Criterion to finish the optimization:
  - 1: Maximal number of function evaluations achieved.

- 2: Maximum allowed CPU time achieved.
- 3: Value to reach achieved.

A.2.6 Guidelines for using *SSm*

Although *SSm* default options have been chosen to be robust for a high number of problems, the tuning of some parameters may help increase the efficiency for a particular problem. Here is presented a list of suggestions for parameter choice depending on the type of problem the user has to face.

- If the problem is likely to be convex, an early local search can find the optimum in short time. For that it is recommended to set the parameter **opts.local.n1 = 0**. Besides, setting **opts.local.n2 = 0** too, the algorithm increases the local search frequency, becoming an "intelligent" multistart.

- When the bounds differ in several orders of magnitude, as is the case of many parameter estimation problems, the distance filter based on Euclidean distances might be inefficient. Choosing the tolerances-based distance filter (i.e., **opts.diverse_criteria = 2**) may help explore different parts of the search space. In those cases, the user should rely on a powerful local search to obtain refined solutions. Also, decision variables indexes may be included in **log_var**.

- For problems with discontinuities and/or noise, the local search should either be deactivated or performed by a direct search method. In those cases, activating the option **opts.local.bestx = 1** may help reduce the computation time wasted in useless local searches, performing one every time the search goes into a new basin of attraction.

- When the function values are very high in absolute value, the weight (**opts.weight**) should be increased to be at least 3 orders of magnitude higher than the mean function value of the solutions found.

- When the search space is very big compared to the area in which the global solution may be located, a first investment in diversification may be useful. For that, a high value of **opts.ndiverse** can help finding good initial solutions to create the initial *RefSet*. A preliminary run with aggressive options can locate a set of good initial solutions for a subsequent optimization with more robust settings. This aggressive search can be

performed by reducing the size of the *RefSet* (**opts.dim_refset**), setting a high function tolerance for joining the *RefSet* (**opts.tolf**), and setting **opts.delete = 'aggressive'**. A more robust search is produced increasing the *RefSet* size.

- When the solutions are very scattered and the best solution in *RefSet* (*fbest*) is close to the global optimum but does not improve as fast as we wish, the intensification frequency (**opts.intens**) can be increased to create solutions close to the best one.
- If local searches are very time-consuming, their tolerance can be relaxed by reducing the value of **opts.local.tol** not to spend a long time in local solution refinements.
- When there are many local solutions close to the global one, the distance filter for the local search may be deactivated (**opts.local.distance_filter = 0**) or the relaxation should be higher by decreasing **opts.local.maxdistfactor**.

In order to reduce the complexity associated to the high number of parameters included in *SSm*, three different basic strategies can be chosen by just adjusting the option **opts.strategy**. Setting this option cancels all the option setting that could have previously been made and uses pre-defined sets of options to perform different types of search. **opts.strategy** may have the three following numerical values:

1. Efficient and fast search. Recommended for unimodal problems or problems in which we need a fast solution (in terms of CPU time or number of function evaluations).
2. Default *SSm* options. It offers a compromise between intensification and diversification and is recommended for most of the problems. Note that choosing this options cancels other possible options previously defined by the user.
3. Robust search. Recommended for searching in different areas of the search space. This strategy sacrifices the speed of the convergence to the global optimum by searching in different areas.

A.3 Extra tools

A.3.1 *ssm_multistart*

This tool allows the user to perform a multistart optimization procedure with any of the local solvers implemented in the *SSm* toolbox using the same problem declaration as with *SSm*. The script `ssm_multistart` has the same input arguments as `ssm_kernel`.

```
>> Results_multistart=multistart(problem,opts)
```

The structure **problem** has the same fields as in *SSm* (except **problem.vtr** which does not apply here). The structure **opts** has only a few fields compared with *SSm* (i.e., **opts.ndiverse**, **opts.local.solver**, **opts.local.tol** and **opts.local.iterprint**). They all work like in *SSm* except **opts.ndiverse**, which indicates the number of initial points chosen for the multistart procedure. A histogram with the final solutions obtained and their frequency is presented at the end of the procedure.

The output structure **Results_multistart** contains the following fields:

- **.fbest**: Best objective function value found after the multistart optimization.
- **.xbest**: Vector providing the best function value.
- **.x0**: Matrix containing the vectors used for the multistart optimization.
- **.f0**: Vector containing the objective function values of the vectors in **Results_multistart.x0**.
- **.func**: Vector containing the objective function values obtained after every local search.
- **.xxx**: Matrix containing the vectors provided by the local optimizations.
- **.no_conv**: Matrix containing the initial points that did not converge to any solution.
- **.nfuneval**: Matrix containing the number of function evaluations performed in every optimization.

A.3.2 *ssm_test*

This tool allows the user to perform a number of optimizations using *SSm* for different problems. It is useful to test the performance of *SSm* with a problem using different sets of options or to check the same set of options with different problems.

```
>> ssm_test(nproblem,noptim,pnames,lb,ub,problem,opts,test,param);
```

The following input parameters must be defined:

- **nproblem**: Number of problems to be tested (if we are testing the same problem n times, set `nproblem = n`, not 1).
- **noptim**: Number of optimizations to perform per problem.
- **pames**: Cell array containing the names of the problems as strings.

- **lb**: Cell array containing the lower bounds for all problems.
- **ub**: Cell array containing the upper bounds for all problems.
- **problem**: Matrix and structure containing problem settings for each problem. These settings are declared like in *SSm* but using indexes. For example, if we want to set an initial point for problem 3 we would type `problem(3).x_0=[x_0_1 x_0_2, ..., x_0_n]`.
- **opts**: General options for all problems. They are declared exactly the same way as in *SSm*.
- **test**: Matrix and structure declaring specific options for individual problems.
- **param**: Structure declaring extra input parameters to be passed to every problem.

ssm_test generates two output .mat files, called **Results_testssm_XXX.mat** and **testsummary_XXX.mat** (where XXX is a number related to the date and time when the test was performed), containing the following variables:

- **Results_testssm_XXX.mat**: Problem settings, options and results (with the same outputs as in *SSm*) for each run under the format $\mathbf{prob\_p_x}$, $\mathbf{opts\_p_x}$ and $\mathbf{res\_p_x\_r_y}$ respectively, where $\mathbf{x}$ is the problem number and $\mathbf{y}$ is the run number.
- **testsummary_XXX.mat**: Summary of some results:
  - $\mathbf{fbest\_p_x}$: Vector containing the best value found in each run for problem $\mathbf{x}$.
  - $\mathbf{neval\_p_x}$: Vector containing the number of evaluations in each run for problem $\mathbf{x}$.
  - $\mathbf{time\_p_x}$: Vector containing the CPU time consumed in each run for problem $\mathbf{x}$.
  - **best_values**: Vector containing the best function value found for each problem after all the runs.
  - **worst_values**: Vector containing the worst function value found for each problem after all the runs.
  - **mean_values**: Vector containing the mean function value found for each problem after all the runs.

A.4 Application examples

A.4.1 Unconstrained problem

$$\min_x f(x) = x_1^2 - 2.1x_1^4 + 1/3x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

subject to

$$-1 \leq x_1, x_2 \leq 1$$

The objective function is defined in `ex1.m`. Note that being an unconstrained problem, there is only one output argument, f.

ex1.m script

```
function f=ex1(x)
f=4*x(1).*x(1)-2.1*x(1).^4+1/3*x(1).^6+x(1).*x(2)-4*x(2).*x(2)+4*x(2).^4;
return
```

The solver is called in `main_ex1.m`. This problem has two known global optima in $x^* = (0.0898, -0.7127)$ and $x^* = (-0.0898, 0.7127)$ with $f(x^*) = -1.03163$.
Options set:

- Maximum number of function evaluations set to 500.
- Maximum number of initial diverse solutions set to 40.
- Local solver chosen: *solnp*.
- Local solver for final refinement: *fmincon*.
- Show the information provided by local solvers on screen.

A.4.2 Constrained problem

$$\min_x f(x) = -x_1 - x_2$$

main_ex1.m script

```
%======================== PROBLEM SPECIFICATIONS ===========================
problem.f='ex1';                        %mfile containing the objective function
problem.x_L=-1*ones(1,2);               %lower bounds
problem.x_U=ones(1,2);                  %upper bounds

opts.maxeval=500;
opts.ndiverse=40;
opts.local.solver='solnp';
opts.local.finish='fmincon';
opts.local.iterprint=1;
%======================== END OF PROBLEM SPECIFICATIONS ====================

Resuls=ssm_kernel(problem,opts);
```

subject to

$$x_2 \leq 2x_1^4 - 8x_1^3 + 8x_1^2 + 2$$

$$x2 \leq 4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1 + 36$$

$$0 \leq x_1 \leq 3$$

$$0 \leq x_1 \leq 4$$

The objective function is defined in `ex2.m`. Note that being a constrained problem, there are two output argument, f and g.

ex2.m script

```
function [f,g]=ex2(x)
f=-x(1)-x(2);
g(1)=x(2)-2*x(1).^4+8*x(1).^3-8*x(1).^2;
g(2)=x(2)-4*x(1).^4+32*x(1).^3-88*x(1).^2+96*x(1);
return
```

The solver is called in `main_ex2.m`. The global optimum for this problem is located in $x^* = [2.32952, 3.17849]$ with $f(x^*) = -5.50801$.

Options set:

- Maximum number of function evaluations set to 750.
- Increase frequency of local solver calls. The first time the solver is called after 100 function evaluations. From that moment, a minimum number of 20 function evaluations will be performed between two consecutive local searches.

main_ex2.m script

```
%======================== PROBLEM SPECIFICATIONS ===========================
problem.f='ex2';                          %mfile containing the objective function
problem.x_L=[0 0];                        %lower bounds
problem.x_U=[3 4];                        %upper bounds
problem.c_L=[-inf -inf];
problem.c_U=[2 36];

opts.maxeval=750;
opts.local.n1=100;
opts.local.n2=20;
%======================== END OF PROBLEM SPECIFICATIONS ====================

Results=ssm_kernel(problem,opts);
```

A.4.3 Constrained problem with equality constraints

$$\min_x f(x) = -x_4$$

subject to

$$\begin{aligned} x_4 - x_3 + x_2 - x_1 + k_4 x_4 x_6 &= 0 \\ x_1 - 1 + k_1 x_1 x_5 &= 0 \\ x_2 - x_1 + k_2 x_2 x_6 &= 0 \\ x_3 + x_1 - 1 + k_3 x_3 x_5 &= 0 \\ x_5^{0.5} + x_6^{0.5} &\leq 4 \\ 0 \leq x_1, x_2, x_3, x_4 &\leq 1 \\ 0 \leq x_5, x_6 &\leq 16 \end{aligned}$$

with $k_1 = 0.09755988$, $k_3 = 0.0391908$ $k_2 = 0.99k_1$ and $k_4 = 0.9k_3$. The objective function is defined in `ex3.m`. Note that equality constraints must be declared before inequality constraints. Parameters $k_1, \ldots, k_4$ are passed to the objective function through the main script, therefore they do not have to be calculated in every function evaluation. See the input arguments below.

The solver is called in `main_ex3.m`. The global optimum for this problem is located in $x^* = [0.77152, 0.516994, 0.204189, 0.388811, 3.0355, 5.0973]$ with $f(x^*) = -0.388811$.

Options set:

ex3.m script

```
function [f,g]=ex3(x,k1,k2,k3,k4)
f=-x(4);

%Equality constraints
g(1)=x(4)-x(3)+x(2)-x(1)+k4*x(4).*x(6);
g(2)=x(1)-1+k1*x(1).*x(5);
g(3)=x(2)-x(1)+k2*x(2).*x(6);
g(4)=x(3)+x(1)-1+k3*x(3).*x(5);

%Inequality constraint
g(5)=x(5).^0.5+x(6).^0.5;

return
```

- Number of equality constraints set to 4 in `problem.neq`.
- Fields `problem.c_L` and `problem.c_U` only contain bounds for inequality constraints.
- Maximum computation time set to 7 seconds.
- Local solver chosen: *solnp*.
- Parameters $k_1, \ldots, k_4$ are passed to the main routine as input arguments.

main_ex3.m script

```
%======================== PROBLEM SPECIFICATIONS ===========================
problem.f='ex3';
problem.x_L=[0 0 0 0 0 0];
problem.x_U=[1 1 1 1 16 16];
problem.neq=4;
problem.c_L=-inf;
problem.c_U=4;

opts.maxeval=1e7;
opts.maxtime=7;
opts.local.solver='solnp';

%======================== END OF PROBLEM SPECIFICATIONS ====================
k1=0.09755988;
k3=0.0391908;
k2=0.99*k1;
k4=0.9*k3;

[Results]=ssm_kernel(problem,opts,k1,k2,k3,k4);;
```

A.4.4 Mixed integer problem

$$\min_x f(x) = x_2^2 + x_3^2 + 2x_1^2 + x_4^2 - 5x_2 - 5x_3 - 21x_1 + 7x_4$$

subject to

$$x_2^2 + x_3^2 + x_1^2 + x_4^2 + x_2 - x_3 + x_1 - x_4 \leq 8$$
$$x_2^2 + 2x_3^2 + x_1^2 + 2x_4^2 - x_2 - x_4 \leq 10$$
$$2x_2^2 + x_3^2 + x_1^2 + 2x_2 - x_3 - x_4 \leq 5$$

Integer variables: x_2, x_3 and x_4. In the function declaration (`ex4.m`) they must have the last indexes.

ex4.m script

```
function [f,g]=ex4(x)
f = x(2)^2 + x(3)^2 + 2*x(1)^2 + x(4)^2 - 5*x(2) - 5*x(3) - 21*x(1) + 7*x(4);
g(1) = x(2)^2 + x(3)^2 + x(1)^2 + x(4)^2 + x(2) - x(3) + x(1) - x(4);
g(2) = x(2)^2 + 2*x(3)^2 + x(1)^2 + 2*x(4)^2 - x(2) - x(4);
g(3) = 2*x(2)^2 + x(3)^2 + x(1)^2 + 2*x(2) - x(3) - x(4);
return
```

The solver is called in `main_ex4.m`. The global optimum for this problem is located in $x^* = [2.23607, 0, 1, 0]$ with $f(x^*) = -40.9575$.
Options set:

- An initial point is specified.
- The number of integer variables is specified (mandatory).
- For mixed integer problems the only solver available is *misqp*.
- Increase the number of function evaluations so that the stop criterion is determined by the CPU time (30 seconds).

A.4.5 Dynamic parameter estimation problem using *n2fb*

Here we will illustrate the use of *SSm* using *n2fb* as local solver. In particular, the problem considered is the isomerization of α-pinene (Section 7.1). In order to use *n2fb* as local solver, in the script `alfa_pinene.m` there must be three output arguments: apart from the objective

main_ex4.m script

```
%======================== PROBLEM SPECIFICATIONS ===========================
problem.f='ex4';
problem.x_L=[0 0 0 0];
problem.x_U=[10 10 10 10];
problem.x_0=[3 4 5 1];
problem.int_var=3;
problem.c_L=[-inf -inf -inf ];
problem.c_U=[8 10 5];


opts.maxeval=1e7;
opts.maxtime=30;
opts.local.solver='misqp';

%======================== END OF PROBLEM SPECIFICATIONS ====================
Results=ssm_kernel(problem,opts);
```

function and the constraints (empty in this case), a vector R containing the squares of the residuals must be defined.

alfa_pinene.m script

```
function [f,g,R] = alfa_pinene(x,t,yexp);

%Integration of the ODE's, providing yteor


%Objective function formulated in the right way for n2fb
f = sum(sum((yteor-yexp).^2));
g=[];

%Build the vector R, needed for using n2fb
R=(yteor-yexp);
R=reshape(R,numel(R),1);

return
```

The solver is called in `main_alfa_pinene.m`

Options set:

- An initial point is specified.
- All the variables are declared as *log_var*.

A.4.6 *ssm_multistart* application

An application of *ssm_multistart* with the problem *ex3* using *solnp* as local solver is presented in the script `main_multistart_ex3.m`. The number of initial points chosen is 25.

main_alfa_pinene.m script

```
%======================== PROBLEM SPECIFICATIONS ==========================
problem.f='alfa_pinene';

problem.x_L=zeros(1,5);
problem.x_U=ones(1,5);

problem.x_0=0.5*ones(1,5);

opts.maxtime=1e6;

opts.maxeval=1e4;

opts.log_var=[1:5];

opts.local.solver='n2fb';
%======================== END OF PROBLEM SPECIFICATIONS ====================
%time intervals

t=[0.0 1230.0 3060.0 4920.0 7800.0 10680.0 15030.0 22620.0 36420.0];

% Distribution of species concentration
 %      y(1)    y(2)    y(3)    y(4)    y(5)

 yexp=[ 100.0   0.0   0.0     0.0     0.0
  88.35    7.3     2.3     0.4     1.75
        76.4    15.6     4.5     0.7     2.8
        65.1    23.1     5.3     1.1     5.8
        50.4    32.9     6.0     1.5     9.3
        37.5    42.7     6.0     1.9    12.0
        25.9    49.1     5.9     2.2    17.0
        14.0    57.4     5.1     2.6    21.0
         4.5    63.1     3.8     2.9    25.7 ];

Results=ssm_kernel(problem,opts,t,yexp);
```

A.4.7 *test_ssm* application

This example will perform a test for problems *ex1*-*ex5* described above. The code for doing this test is presented in the script `main_test.m`

main_multistart_ex3.m script

```
%======================== PROBLEM SPECIFICATIONS ==========================
problem.f='ex3';
problem.x_L=[0 0 0 0 0 0];
problem.x_U=[1 1 1 1 16 16];
problem.neq=4;
problem.c_L=-inf;
problem.c_U=4;

opts.ndiverse=25;

opts.local.solver='solnp';
opts.local.iterprint=1;
opts.local.tol=3;
%==========================================================================
k1=0.09755988;k3=0.0391908;k2=0.99*k1;k4=0.9*k3;

Results_multistart=ssm_multistart(problem,opts,k1,k2,k3,k4);
```

main_test.m script

```
nproblem=5;                                     %Number of problems to be tested
noptim=1;                                       %Number of optimization per problem
pnames={'ex1', 'ex2', 'ex3', 'ex4','ex5'};      %Name of all problems

%Lower and upper bounds for all problems
lb={-1*ones(1,2), [0 0], [0 0 0 0 0 0], [0 0 0 0], zeros(1,5)};
ub={ones(1,2), [3 4], [1 1 1 1 16 16], [10 10 10 10], ones(1,5)};

%Specific problem settings
%Problem 1
problem(1).vtr=-1.031628;        %Value to reach for problem 1

%Problem 2
problem(2).c_L=[-inf -inf];      %Lower bounds for problem 2 nonlinear inequality constraints
problem(2).c_U=[2 36];           %Upper bounds for problem 2 nonlinear inequality constraints

%Problem 3
problem(3).neq=4;                %Number of nonlinear equality constraints in problem 3
problem(3).c_L=-inf;             %Lower bounds for problem 3 nonlinear inequality constraints
problem(3).c_U=4;                %Upper bounds for problem 3 nonlinear inequality constraints
problem(3).vtr=-0.3888;          %Value to reach for problem 3

%Problem 4
problem(4).x_0=[3 4 5 1];        %Initial point for problem 4
problem(4).int_var=3;            %Number of integer variables in problem 4
problem(4).c_L=[-inf -inf -inf ];%Lower bounds for problem 4 nonlinear inequality constraints
problem(4).c_U=[8 10 5];         %Upper bounds for problem 4 nonlinear inequality constraints

%Problem 5
problem(5).vtr=19.8722;          %Value to reach for problem 5

%Options for all problems
opts.maxeval=1e5;                %Maximum number of function evaluations for all problems
opts.maxtime=3;                  %Maximum computation time for all problems

%Specific options for some problems
test(3).local.solver='solnp';    %Specific local solver for problem 3
test(4).local.solver='misqp';    %Specific local solver for problem 4
test(5).maxtime=100;             %Increase the optimization time for problem 5
test(5).log_var=[1:5];           %Declare all variables as log_var for problem 5
test(5).local.solver='n2fb';     %Specific local solver for problem 5

%Extra input parameters for some problems
%Problem 3
k1=0.09755988; k3=0.0391908; k2=0.99*k1; k4=0.9*k3;
param{3}={k1,k2,k3,k4};

%Problem5
%time intervals
t=[0.0 1230.0 3060.0 4920.0 7800.0 10680.0 15030.0 22620.0 36420.0];
% Distribution of species concentration
 %      y(1)    y(2)    y(3)    y(4)    y(5)
 yexp=[ 100.0  0.0  0.0  0.0     0.0
     88.35    7.3     2.3     0.4     1.75
        76.4    15.6     4.5     0.7     2.8
        65.1    23.1     5.3     1.1     5.8
        50.4    32.9     6.0     1.5     9.3
        37.5    42.7     6.0     1.9    12.0
        25.9    49.1     5.9     2.2    17.0
        14.0    57.4     5.1     2.6    21.0
         4.5    63.1     3.8     2.9    25.7 ];
param{5}={t,yexp};
%Call testssm
ssm_test(nproblem,noptim,pnames,lb,ub,problem,opts,test,param);
```

A.5 Help files

A.5.1 *SSm* help file

```
%Function   : SSm beta 3.3
%Written by : Process Engineering Group IIM-CSIC (jegea@iim.csic.es)
%Created on : 15/06/2005
%Last Update: 03/03/2008
%
%Global optimization algorithm for MINLP's based on Scatter Search
%
%   SSm attempts to solve problems of the form:
%       min F(x)  subject to:  ceq(x) = 0 (equality constraints)
%        x                     c_L <= c(x) <= c_U (inequality constraints)
%                              x_L <= x <= x_U (bounds on the decision variables)
%
%Please have a look at the manual before using SSm
%
%       USAGE: Results = ssm_kernel(problem,opts,p1,p2,....,pn);
%
%INPUT PARAMETERS:
%****************
%   problem - Structure containing problem settings
%       problem.f   = Name of the file containing the objective
%                     function
%       problem.x_L = Lower bounds of decision variables
%       problem.x_U = Upper bounds of decision variables
%       problem.x_0 = Initial point(s) (optional)
%
%       Additionally, fill the following fields if your problem has
%       non-linear constraints
%       problem.neq     = Number of equality constraints (do not define it
%                         if there are no equality constraints)
%       problem.c_L     = Lower bounds of nonlinear inequality constraints
%       problem.c_U     = Upper bounds of nonlinear inequality constraints
%       problem.int_var = Number of integer variables
%       problem.bin_var = Number of binary variables
%       problem.vtr     = Objective function value to be reached (optional)
%
%NOTE: The order of decision variables is x=[cont int bin]
%
%   opts - Structure containing options (if set as opts=[] defaults options
%          will be loaded) Type "ssm_kernel" or "ssm_kernel('defaults')" to
%          get the default options
%
%       User options
%           opts.maxeval    = Maximum number of function evaluations
%                             (Default 1000)
%           opts.maxtime    = Maximum CPU time in seconds (Default 60)
%           opts.iterprint  = Print each iteration on screen: 0-Deactivated
%                             1-Activated (Default 1)
%           opts.plot       = Plots convergence curves: 0-Deactivated,
%                             1-Plot curves on line, 2-Plot final results
%                             (Default 0)
%           opts.weight     = Weight that multiplies the penalty term added
%                             to the objective function in constrained
%                             problems (Default 1000)
%           opts.log_var    = Indexes of the variables which will be used
%                             to generate diverse solutions in different
```

```
%                             orders of magnitude
%          opts.tolc       = Maximum absolute violation of the constraints
%                             (Default 1e-5)
%          opts.save_report= Saves Results, problem and opts in a .mat
%                             file (Default 0)
%          opts.report_name= Report name (Default 'ssm_report.mat')
%
%      Global options
%          opts.dim_refset         = Number of elements in Refset
%                                    (automatically calculated)
%          opts.ndiverse           = Number of solutions generated by the
%                                    diversificator (Default 10*nvar)
%          opts.initiate           = Type of Refset initialization
%                                    (Default 0)
%                                    0: Take bounds, middle point and fill
%                                    by euclidean distance
%                                    1: Evaluate all the diverse
%                                    solutions,take the dim_refset/2 best
%                                    solutions and fill by euclidean
%                                    distance
%          opts.combination        = Type of combination of Refset
%                                    elements (Default 1)
%                                    1: hyper-rectangles
%                                    2: linear combinations
%          opts.regenerate         = Type of Refset regeneration (Default
%                                    3)
%                                    1: Regeneration by distance diversity
%                                    2: Regeneration by direction
%                                    diversity
%                                    3: Randomly alternates 1 and 2
%          opts.delete             = Maximum number of Refset elements
%                                    deleted when regenerating Refset
%                                    (Default 'standard')
%                                    'standard': Maximum deleted elements=
%                                    dim_refset/2 (half of the elements)
%                                    'aggressive': Delete dim_refset-1
%                                    (all of them except the best solution
%                                    found)
%          opts.intens             = Iteration interval between
%                                    intensifications (default 10)
%          opts.tolf               = Function tolerance for joining the
%                                    Refset (default 1e-4)
%          opts.diverse_criteria   = Criterion for diversification in the
%                                    Refset (Default 1)
%                                    1: euclidean distance
%                                    2: tolerances
%          opts.tolx               = Variable tolerance for joining the
%                                    Refset when the euclidean distance is
%                                    deactivated(default 1e-3 for all
%                                    variables)
%      Local options
%          opts.local.solver              = Choose local solver
%                                           0: Local search deactivated
%                                           'fmincon'(Default),'constrnew'
%                                           'fminsearch','nomad', 'solnp'
%                                           'n2fb','dn2fb','dhc','fsqp'
%                                           'ipopt','misqp','lsqnonlin'
%          opts.local.tol                 = Level of tolerance in local
%                                           search
```

```
%           opts.local.iterprint            = Print each iteration of local
%                                             solver on screen
%           opts.local.n1                   = Number of function
%                                             evaluations before applying
%                                             local search for the 1st time
%                                             (Default 100*nvar)
%           opts.local.n2                   = Minimum number of function
%                                             evaluations in the global
%                                             phase between 2 local calls
%                                             (Default 200*nvar)
%           opts.local.finish               = Applies local search to the
%                                             best solution found once the
%                                             optimization if finished
%                                             (same values as
%                                             opts.local.solver)
%           opts.local.bestx                = When activated (i.e. =1) only
%                                             applies local search to the
%                                             best solution found to
%                                             date,ignoring filters
%                                             (Default=0)
%           opts.local.merit_filter         = Activation of merit filter
%                                             for local search (Default 1)
%                                             0: Filter deactivated
%                                             1: Filter activated
%           opts.local.distance_filter      = Activation of distance filter
%                                             for local search (Default 1)
%                                             0: Filter deactivated
%                                             1: Filter activated
%           opts.local.thfactor             = Merit filter relaxation
%                                             parameter (Default 0.2)
%           opts.local.maxdistfactor        = Distance filter relaxation
%                                             parameter (Default 0.2)
%           opts.local.wait_maxdist_limit   = Apply distance filter
%                                             relaxation after this number
%                                             of function evaluations
%                                             without success in passing
%                                             filter (Default 20)
%           opts.local.wait_th_limit        = Apply merit filter relaxation
%                                             after this number of function
%                                             evaluations without success in
%                                             passing filter (Default 20)
%
%           opts.strategy                   = If >0, it selects different
%                                             set of options to perform
%                                             different types of searches
%                                             1: Fast and efficient
%                                             2: Average (default options)
%                                             3: Robust
%
%
%   p1,p2... :  optional input parameters to be passed to the objective
%   function
%
%
%OUTPUT PARAMETERS:
%*****************
% A file called "ssm_report.mat" is generated containing.
%
%   problem - Structure containing problem settings
```

```
%   opts    - Structure containing all options
%   Results - Structure containing results
%
% Fields in Results
%       Results.fbest                   = Best objective function value
%                                         found after the optimization
%       Results.xbest                   = Vector providing the best
%                                         function value
%       Results.cpu_time                = Time in seconds consumed in the
%                                         optimization
%       Results.f                       = Vector containing the best
%                                         objective function value after each
%                                         iteration
%       Results.x                       = Matrix containing the best vector
%                                         in each iteration
%       Results.time                    = Vector containing the cpu time
%                                         consumed after each iteration
%       Results.neval                   = Vector containing the number of
%                                         function evaluations after each
%                                         iteration
%       Results.numeval                 = Number of function evaluations
%       Results.local_solutions         = Local solutions found by the
%                                         local solver (in rows)
%       Results.local_solutions_values  = Function values of the local
%                                         solutions
%       Results.end_crit                = Criterion to finish the
%                                         optimization
%                                         1: Maximal number of function
%                                         evaluations achieved
%                                         2: Maximum allowed CPU Time
%                                         achieved
%                                         3: Value to reach achieved
%
%              NOTE: To plot convergence curves type:
%              stairs(Results.time,Results.f) or stairs(Results.neval,Results.f)
```

A.5.2 *SSKm* help file

```
%Function   : SSkm lite beta
%Written by : Process Engineering Group IIM-CSIC (jegea@iim.csic.es)
%Created on : 11/09/2007
%
%Global Optimization Algorithm for unconstrained costly continuous problems
%based on Scatter Search and Kriging.
%
%   SSm attempts to solve problems of the form:
%       min F(x)  subject to: x_L <= x <= x_U (bounds on the decision variables)
%
%   USAGE: [fbest,xbest,cpu_time]=sskm_lite(problem,opts);
%
%INPUT PARAMETERS:
%*****************
%   problem - Structure containing problem
%       problem.f   = Name of the file containing the objective function
%       problem.x_L = Lower bounds of decision variables
%       problem.x_U = Upper bounds of decision variables
%       problem.x_0 = Initial point(s) (optional)
%       problem.f_0 = Function values of initial point(s) (optional)
```

```
%       problem.vtr = Objective function value to be reached (optional)
%
%       NOTE:The dimension of f_0 and x_0 maybe different. For example, if
%       we want to introduce 5 initial points but we only know the values
%       for 3 of them, x_0 would have 5 rows whereas f_0 would have only 3
%       elements. It is mandatory that the first 3 rows of x_0 correspond
%       to the values of f_0
%       If your problem has constraints please introduce them as a penalty
%       term in the objective function
%
%   opts - Structure containing options (if not defined, defaults options
%   will be loaded)
%       User options
%          opts.maxeval = Maximum number of function evaluations (Default 1000)
%          opts.maxtime = Maximum CPU time in seconds (Default 60)
%          opts.plot    = Plots convergence curves 0-Deactivated 1-Plot
%                         convergence curves (Default 0)
%          opts.log_x   = Vector containing the indexes of the variables
%                         whose logarithm will be used to make the
%                         surrogate surface smoother and to search in
%                         differents orders of magnitude. Only possible if
%                         the decision variables are always positive.
%           opts.log_f  = Takes the logarithm of the function values to
%                         make the surrogate surface smoother. Only
%                         possible if the function values are always
%                         positive. 0-Deactivated (default), 1-Activated
%
%       Global options
%           opts.dim_refset     = Number of elements in Refset (Default 10)
%           opts.ndiverse       = Number of solutions generated by the
%                                 diversificator (Default 100)
%           opts.combination    = Type of combination of Refset elements
%                                 1-pseudo-linear combinations (default),
%                                 2: linear combinations
%
%OUTPUT PARAMETERS
%*****************
%   Results - Structure containing results
%       Results.fbest       = Best objective function value found after the
%                             optimization
%       Results.xbest       = Vector providing the best function value
%       Results.cpu_time    = Time in seconds consumed in the optimization
%       Results.x           = Matrix containing the vectors that provided
%                             successful simulations
%       Results.crash       = Matrix containing the vectors that provided a
%                             simulation error
%       Results.func        = Vector containing all the function
%                             evaluations
%       Results.time        = Vector containing the cpu time consumed after
%                             each evaluation
%       Results.f           = Vector containing the best objective function
%                             value found
%       Results.numeval     = Number of function evaluations
%       Results.numeval_ss  = Number of function evaluations done by
%                             Scatter Search in the same problem
%       Results.end_crit    = Criterion to finish the optimization
%                             1: Maximal number of function evaluations achieved
%                             2: Maximal computation time achieved
%                             3: "Value to reach" achieved
```

```
%
%NOTE: To plot convergence curves type: stairs(Results.time,Results.f) or
%      stairs(1:Results.numeval,Results.f)
```

Appendix B

Test Functions of Section 4.3

B.1 Unconstrained problems

1. Branin

Minimize $f(\mathbf{x}) = \left(x_2 - \left(\frac{5}{4\pi^2}\right) x_1^2 + \left(\frac{5}{\pi}\right) x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right) \cos(x_i) + 10$

Subject to $-5 \leq x_i, x_2 \leq 15$ for $i = 1, 2$

2. B2

Minimize $f(\mathbf{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$

Subject to $-50 \leq x_i \leq 100$ for $i = 1, 2$

3. Easom

Minimize $f(\mathbf{x}) = -\cos(x_1)\cos(x_2)\exp\left(-\left((x_1 - \pi)^2 + (x_2 - \pi)^2\right)\right)$

Subject to $-100 \leq x_i \leq 100$ for $i = 1, 2$

4. Goldstein and Price

$$\text{Minimize} \quad f(\mathbf{x}) = \begin{array}{l}\left(1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right) \\ \left(30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right)\end{array}$$

Subject to $-2 \leq x_i \leq 2$ for $i = 1, 2$

5. Shubert

Minimize $f(\mathbf{x}) = \left(\sum_{j=1}^{5} j\cos\left((j+1)x_1 + j\right)\right)\left(\sum_{j=1}^{5} j\cos\left((j+1)x_2 + j\right)\right)$

Subject to $-10 \leq x_i \leq 10$ for $i = 1, 2$

6. Beale

Minimize $f(\mathbf{x}) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$

Subject to $-4.5 \leq x_i \leq 4.5$ for $i = 1, 2$

7. Booth

Minimize $f(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$

Subject to $-10 \leq x_i \leq 10$ for $i = 1, 2$

8. Matyas

Minimize $f(\mathbf{x}) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$

Subject to $-5 \leq x_i \leq 10$ for $i = 1, 2$

9. Six Hump Camel Back

Minimize $f(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$

Subject to $-5 \leq x_i \leq 5$ for $i = 1, 2$

10, 23. Schwefel(n)

Minimize $f(\mathbf{x}) = -\sum_{i=1}^{n} \left(-x_i \sin\sqrt{|x_i|}\right)$

Subject to $-500 \leq x_i \leq 500$ for $i = 1, \ldots, n$

11, 29, 34. Rosenbrock(n)

Minimize $f(\mathbf{x}) = \sum_{i=1}^{n/2} 100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2$

Subject to $-10 \leq x_i \leq 10$ for $i = 1, \ldots, n$

12, 30, 35. Zakharov(n)

Minimize $f(\mathbf{x}) = \sum_{j=1}^{n} x_j^2 + \left(\sum_{j=1}^{n} 0.5jx_j\right)^2 + \left(\sum_{j=1}^{n} 0.5jx_j\right)^4$

Subject to $-5 \leq x_i \leq 10$ for $i = 1, \ldots, n$

13. De Joung

Minimize $f(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2$

Subject to $-2.56 \leq x_i \leq 5.12$ for $i = 1, 2, 3$

14. Hartman(3,4)

Minimize $f(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right)$

Subject to $0 \leq x_i \leq 1$ for $i = 1, 2, 3$

i	a_{ij}			c_i	p_{ij}		
1	3.0	10.0	30.0	1.0	0.3689	0.1170	0.2673
2	0.1	10.0	35.0	1.2	0.4699	0.4387	0.7470
3	3.0	10.0	30.0	3.0	0.1091	0.8732	0.5547
4	0.1	10.0	35.0	3.2	0.0381	0.5743	0.8828

15. Colville

Minimize $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1\left((x_2 - 1)^2 + (x_4 - 1)^2\right) + 19.8(x_2 - 1)(x_4 - 1)$

Subject to $-10 \leq x_i \leq 10$ for $i = 1, \ldots, 4$

16-18. Shekel(n)

Minimize $f(\mathbf{x}) = -\sum_{i=1}^{n} \left((\mathbf{x} - \mathbf{a_i})^T (\mathbf{x} - \mathbf{a_i}) + c_i\right)^{-1}$; $\mathbf{x} = (x_1, x_2, x_3, x_4)^T$; $\mathbf{a}_i = \left(a_i^1, a_i^2, a_i^3, a_i^4\right)^T$

Subject to $0 \leq x_i \leq 10$ for $i = 1, \ldots, 4$

i	a_i^T				c_i
1	4.0	4.0	4.0	4.0	0.1
2	1.0	1.0	1.0	1.0	0.2
3	8.0	8.0	8.0	8.0	0.2
4	6.0	6.0	6.0	6.0	0.4
5	3.0	7.0	3.0	7.0	0.4
6	2.0	9.0	2.0	9.0	0.6
7	5.0	5.0	3.0	3.0	0.3
8	8.0	1.0	8.0	1.0	0.7
9	6.0	2.0	6.0	2.0	0.5
10	7.0	3.6	7.0	3.6	0.5

19. Perm(n, β)

Minimize $f(\mathbf{x}) = \sum_{k=1}^{n} \left(\sum_{i=1}^{n} \left(i^k + \beta\right) \left(\left(\frac{x_i}{i}\right)^k - 1\right)\right)^2$

Subject to $-n \leq x_i \leq n$ for $i = 1, \ldots, n$

20. Perm0(n, β)

Minimize $f(\mathbf{x}) = \sum_{k=1}^{n} \left(\sum_{i=1}^{n} (i + \beta) \left(x_i^k - \left(\frac{1}{i}\right)^k\right)\right)^2$

Subject to $-n \leq x_i \leq n$ for $i = 1, \ldots, n$

21. PowerSum($b_1, \ldots, b_n$)

Minimize $f(\mathbf{x}) = \sum_{k=1}^{n} \left(\left(\sum_{i=1}^{n} x_i^k \right) - b_k \right)^2$

Subject to $0 \leq x_i \leq n$ for $i = 1, \ldots, n$

22. Hartmann(6,4)

Minimize $f(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij} \left(x_j - p_{ij} \right)^2 \right)$

Subject to $0 \leq x_i \leq 1$ for $i = 1, \ldots, 6$

i	a_{ij}						c_i	p_{ij}					
1	10.0	3.0	17.0	3.5	1.7	8.0	1.0	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10.0	17.0	0.10	8.0	14.0	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3.0	3.5	1.7	10.0	17.0	8.0	3.0	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
4	17.0	8.0	0.05	10.0	0.1	14.0	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

24-25. Trid(n)

Minimize $f(\mathbf{x}) = \left(\sum_{i=1}^{n} \left(x_i - 1 \right)^2 \right) - \sum_{i=2}^{n} x_i x_j$

Subject to $-n^2 \leq x_i \leq n^2$ for $i = 1, \ldots, n$

26-31. Rastrigin(n)

Minimize $f(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos\left(2\pi x_i\right) \right)$

Subject to $-2.56 \leq x_i \leq 5.12$ for $i = 1, \ldots, n$

27-32. Griewank(n)

Minimize $f(\mathbf{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$

Subject to $-300 \leq x_i \leq 600$ for $i = 1, \ldots, n$

28-33. Sum Squares(n)

Minimize $f(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2$

Subject to $-5 \leq x_i \leq 10$ for $i = 1, \ldots, n$

36. Powell(n)

Minimize $f(\mathbf{x}) = \sum_{j=1}^{n/4} (x_{4j-3} + 10x_{4j-2})^2 + 5(x_{4j-1} - x_{4j})^2 + (x_{4j-2} - 2x_{4j-1})^4 + 10(x_{4j-3} - x_{4j})^4$

Subject to $-4 \leq x_i \leq 5$ for $i = 1, \ldots, n$

37. Dixon and Price(n)

Minimize $f(\mathbf{x}) = \sum_{i=1}^{n} i(2x_i^2 - x_{i-1})^2 + (x_1 - 1)^2$

Subject to $-10 \leq x_i \leq 10$ for $i = 1, \ldots, n$

38. Levy(n)

Minimize $f(\mathbf{x}) = \sin^2(\pi y_1) + \sum_{i=1}^{k-1} (y_i - 1)^2 \left(1 + 10\sin^2(\pi y_i + 1)\right) + (y_k - 1)^2(1 + \sin^2(2\pi x_k))$

Subject to $y_i = 1 + \frac{x_i - 1}{4}$ for $i = 1, \ldots, n$
$-10 \leq x_i \leq 10$ for $i = 1, \ldots, n$

39. Sphere(n)

Minimize $f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$

Subject to $-2.56 \leq x_i \leq 5.12$ for $i = 1, \ldots, n$

40. Ackley(n)

Minimize $f(\mathbf{x}) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$

Subject to $-15 \leq x_i \leq 30$ for $i = 1, \ldots, n$

B.2 Constrained problems

1. g01

Minimize $f(\mathbf{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - 5\sum_{i=5}^{1} 3x_i$

Subject to
$g_1(\mathbf{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$
$g_2(\mathbf{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$
$g_3(\mathbf{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$
$g_4(\mathbf{x}) = -8x_1 + x_{10} \leq 0$
$g_5(\mathbf{x}) = -8x_2 + x_{11} \leq 0$
$g_6(\mathbf{x}) = -8x_3 + x_{12} \leq 0$
$g_7(\mathbf{x}) = -2x_4 - x_5 + x_{10} \leq 0$
$g_8(\mathbf{x}) = -2x_6 - x_7 + x_{11} \leq 0$
$g_9(\mathbf{x}) = -2x_8 - x_9 + x_{12} \leq 0$
$0 \leq x_i \leq 1 \;\; (i = 1, \ldots, 9); \;\; 0 \leq x_i \leq 100 \;\; (i = 11, 12, 13); \;\; 0 \leq x_{13} \leq 1$

2. g02

Minimize $f(\mathbf{x}) = -\left|\frac{\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}\cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}}\right|$

$$\text{Subject to} \quad g_1(\mathbf{x}) = 0.75 - \prod_{i=1}^{n} x_i \le 0$$
$$g_2(\mathbf{x}) = \sum_{i=1}^{n} x_i - 7.5n \le 0$$
$$0 \le x_i \le 10 \quad (i = 1, \ldots, 20)$$

3. g03

$$\text{Minimize } f(\mathbf{x}) = -(\sqrt{n})^n \prod_{i=1}^{n} x_i$$

$$\text{Subject to} \quad h_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$
$$0 \le x_i \le 1 \quad (i = 1, \ldots, 10)$$

4. g04

$$\text{Minimize } f(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

$$\text{Subject to} \quad g_1(\mathbf{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0$$
$$g_2(\mathbf{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0$$
$$g_3(\mathbf{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0$$
$$g_4(\mathbf{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0$$
$$g_5(\mathbf{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0$$
$$g_6(\mathbf{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0$$
$$78 \le x_1 \le 102; \quad 33 \le x_2 \le 45; \quad 27 \le x_i \le 45 \quad (i = 3, 4, 5)$$

5. g05

$$\text{Minimize } f(\mathbf{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$$

$$\text{Subject to} \quad g_1(\mathbf{x}) = -x_4 + x_3 - 0.55 \le 0$$
$$g_2(\mathbf{x}) = -x_3 + x_4 - 0.55 \le 0$$
$$h_3(\mathbf{x}) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$
$$h_4(\mathbf{x}) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$
$$h_5(\mathbf{x}) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$
$$0 \le x_1 \le 1200; \quad 0 \le x_2 \le 1200; \quad -0.55 \le x_i \le 0.55 \quad (i = 3, 4)$$

6. g06

$$\text{Minimize } f(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$$

$$\text{Subject to} \quad g_1(\mathbf{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0$$
$$g_2(\mathbf{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0$$
$$13 \le x_1 \le 100; \quad 0 \le x_2 \le 100$$

7. g07

$$\text{Minimize } f(\mathbf{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2$$
$$+2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

$$
\begin{aligned}
\text{Subject to} \quad & g_1(\mathbf{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0 \\
& g_2(\mathbf{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0 \\
& g_3(\mathbf{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0 \\
& g_4(\mathbf{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \\
& g_5(\mathbf{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \\
& g_6(\mathbf{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0 \\
& g_7(\mathbf{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0 \\
& g_8(\mathbf{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0 \\
& -10 \leq x_i \leq 10; \quad (i = 1, \ldots, 10)
\end{aligned}
$$

8. g08

Minimize $f(\mathbf{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$

$$
\begin{aligned}
\text{Subject to} \quad & g_1(\mathbf{x}) = x_1^2 - x_2 + 1 \leq 0 \\
& g_2(\mathbf{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0 \\
& 0 \leq x_i \leq 10; \quad (i = 1, 2)
\end{aligned}
$$

9. g09

$$
\begin{aligned}
\text{Minimize } f(\mathbf{x}) = {} & (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\
& + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7
\end{aligned}
$$

$$
\begin{aligned}
\text{Subject to} \quad & g_1(\mathbf{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0 \\
& g_2(\mathbf{x}) == -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0 \\
& g_3(\mathbf{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0 \\
& g_4(\mathbf{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0 \\
& -10 \leq x_i \leq 10; \quad (i = 1, \ldots, 7)
\end{aligned}
$$

10. g10

Minimize $f(\mathbf{x}) = x_1 + x_2 + x_3$

$$
\begin{aligned}
\text{Subject to} \quad & g_1(\mathbf{x}) = -1 + 0.0025(x_4 + x_6) \leq 0 \\
& g_2(\mathbf{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0 \\
& g_3(\mathbf{x}) = -1 + 0.01(x_8 - x_5) \leq 0 \\
& g_4(\mathbf{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0 \\
& g_5(\mathbf{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0 \\
& g_6(\mathbf{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0 \\
& 100 \leq x_1 \leq 10000; \quad 1000 \leq x_i \leq 10000 \quad (i = 2, 3); \quad 10 \leq x_i \leq 1000 \quad (i = 4, \ldots, 8)
\end{aligned}
$$

11. g11

Minimize $f(\mathbf{x}) = x_1^2 + (x_2 - 1)^2$

$$
\begin{aligned}
\text{Subject to} \quad & h_1(\mathbf{x}) = x_2 - x_1^2 = 0 \\
& -1 \leq x_i \leq 1 \quad (i = 1, 2)
\end{aligned}
$$

12. g12

Minimize $f(\mathbf{x}) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$

Subject to $g(\mathbf{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$
$0 \leq x_i \leq 10 (i = 1, 2, 3); \quad p, q, r = 1, \ldots, 9$

13. g13

Minimize $f(\mathbf{x}) = e^{x_1 x_2 x_3 x_4 x_5}$

Subject to $h_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$
$h_2(\mathbf{x}) = x_2 x_3 - 5 x_4 x_5 = 0$
$h_3(\mathbf{x}) = x1^3 + x_2^3 + 1 = 0$
$-2.3 \leq x_i \leq 2.3 (i = 1, 2); -3.2 \leq x_i \leq 3.2 (i = 3, 4, 5)$

Part VI

Bibliography

Bibliography

Abramson, M. A. (2002). *Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems*. PhD thesis, Rice University.

Adjiman, C. S., Androulakis, I. P., and Floudas, C. A. (1997). Global optimization of MINLP problems in process synthesis and design. *Computers and Chemical Engineering*, 21(SUPPL.1):S445–S450.

Adjiman, C. S., Androulakis, I. P., and Floudas, C. A. (2000). Global optimization of mixed-integer nonlinear problems. *AIChE Journal*, 46(9):1769–1797.

Alasino, N., Mussati, M. C., and Scenna, N. (2007). Wastewater treatment plant synthesis and design. *Industrial and Engineering Chemistry Research*, 46(23):7497–7512.

Alex, J., Beteau, J. F., Copp, J., Hellinga, C., Jeppsson, U., Marsili-Libelli, S., Pons, M. N., Spanjers, H., and Vanhooren, H. (1999). Benchmark for evaluating control strategies in wastewater treatment plants. In *ECC'99 (European Control Conference)*, Karlsruhe (Germany).

Ali, M. M., C., S., and Törn, A. (1997). Application of stochastic global optimization algorithms to practical problems. *Journal of Optimization Theory and Applications*, 95(3):545–563.

Angira, R. and Santosh, A. (2007). Optimization of dynamic systems: A trigonometric differential evolution approach. *Computers and Chemical Engineering*, 31(9):1055–1063.

Asaadi, J. (1973). A computational comparison of some non-linear programs. *Mathematical Programming*, 4(1):144–154.

Auger, A. and Hansen, N. (2005). Performance evaluation of an advanced local search evo-

lutionary algorithm. In *IEEE Congress on Evolutionary Computation, IEEE CEC 2005*, volume 2, pages 1777–1784.

Averick, B. M., Carter, R. G., and Moré, J. J. (1991). The MINPACK-2 test problem collection. Technical Report ANL/MCS-TM-273, Argonne National Laboratory.

Babu, B. V. and Angira, R. (2006). Modified differential evolution (MDE) for optimization of non-linear chemical processes. *Computers and Chemical Engineering*, 30(6-7):989–1002.

Bailey, J. E. (1998). Mathematical modeling and analysis in biochemical engineering: Past accomplishments and future opportunities. *Biotechnology Progress*, 14(1):8–20.

Balku, S. and Berber, R. (2006). Dynamics of an activated sludge process with nitrification and denitrification: Start-up simulation and optimization using evolutionary algorithm. *Computers and Chemical Engineering*, 30(3):490–499.

Balsa-Canto, E., Banga, J. R., Alonso, A. A., and Vassiliadis, V. B. (2004). Dynamic optimization of distributed parameter systems using second-order directional derivatives. *Industrial and Engineering Chemistry Research*, 43(21):6756–6765.

Balsa-Canto, E., Vassiliadis, V. S., and Banga, J. R. (2005). Dynamic optimization of single- and multi-stage systems using a hybrid stochastic-deterministic method. *Industrial and Engineering Chemistry Research*, 44(5):1514–1523.

Banga, J. R., Alonso, A. A., and Singh, R. P. (1997). Stochastic dynamic optimization of batch and semicontinuous bioprocesses. *Biotechnology Progress*, 13(3):326–335.

Banga, J. R., Balsa-Canto, E., Moles, C. G., and Alonso, A. A. (2003a). Dynamic optimization of bioreactors - a review. *Proceedings of the Indian National Science Academy*, 69A(3-4):257–265.

Banga, J. R., Balsa-Canto, E., Moles, C. G., and Alonso, A. A. (2003b). Improving food processing using modern optimization methods. *Trends in Food Science and Technology*, 14(4):131–144.

Banga, J. R., Balsa-Canto, E., Moles, C. G., and Alonso, A. A. (2005). Dynamic optimization of bioprocesses: Efficient and robust numerical strategies. *Journal of Biotechnology*, 117(4):407–419.

Banga, J. R., Moles, C. G., and Alonso, A. A. (2003c). Global optimization of bioprocesses using stochastic and hybrid methods. In Floudas, C. A. and Pardalos, P. M., editors, *Frontiers In Global Optimization*, volume 74 of *Nonconvex Optimization and Its Applications*, pages 45–70. Kluwer Academic Publishers, Hingham, MA, USA.

Banga, J. R., Saa, J., and Alonso, A. A. (1999). Model-based optimization of microwave heating of bioproducts. In *7th International Conference On Microwave and High Frequency Heating*, Valencia (Spain).

Banga, J. R. and Seider, W. D. (1996). Global optimization of chemical processes using stochastic algorithms. In Floudas, C. A. and Pardalos, P. M., editors, *State of the Art in Global Optimization*, pages 563–583. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Banga, J. R. and Singh, R. P. (1994). Optimization of air drying of foods. *Journal of Food Engineering*, 23(2):189–211.

Bansal, V., Perkins, J. D., Pistikopoulos, E. N., Ross, R., and Van Schijndel, J. M. G. (2000). Simultaneous design and control optimisation under uncertainty. *Computers and Chemical Engineering*, 24(2-7):261–266.

Barton, P. I., Banga, J. R., and Galán, S. (2000). Optimization of hybrid discrete/continuous dynamic systems. *Computers and Chemical Engineering*, 24(9-10):2171–2182.

Bellman, R. E. (2003). *Dynamic Programming*. Dover Publications, Incorporated.

Benali, M., Hammache, A., Aubé, F., Dipama, J., and Cantave, R. (2007). Dynamic multi-objective optimization of large-scale industrial production systems: An emerging strategy. *International Journal of Energy Research*, 31(12):1202–1225.

Beyer, H. G. (1996). Toward a Theory of Evolution Strategies: Self-Adaptation. *Evolutionary Computation*, 3(3):311–347.

Beyer, H. G. and Schwefel, H. P. (2002). Evolution strategies - a comprehensive introduction. *Natural Computing*, 1(1):3–52.

Biegler, L. T. and Grossmann, I. E. (2004). Retrospective on optimization. *Computers and Chemical Engineering*, 28(8):1169–1192.

Birattari, M. and Dorigo, M. (2007). How to assess and report the performance of a stochastic algorithm on a benchmark problem: Mean or best result on a number of runs? *Optimization Letters*, 1(3):309–311.

Björkman, M. and Holmström, K. (2000). Global optimization of costly nonconvex functions using radial basis functions. *Optimization Engineering*, 1(4):373–397.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308.

Bogle, I. D. L., Cockshott, A. R., Bulmer, M., Thornhill, N., Gregory, M., and Dehghani, M. (1996). A process systems engineering view of biochemical process operations. *Computers and Chemical Engineering*, 20(6-7):943–949.

Box, G. E. P., Hunter, W. G., MacGregor, J. F., and Erjavec, J. (1973). Some problems associated with the analysis of multiresponse data. *Technometrics*, 15:33–51.

Brooks, S. (1958). A discussion of random methods for seeking maxima. *Operations Research*, 6:244–251.

Bryson, A. E. and Ho, Y.-C. (1975). *Applied optimal control: Optimization, estimation, and control*. Hemisphere Pub. Corp., Washington and New York.

Butler, D. and Schütze, M. (2005). Integrating simulation models with a view to optimal control of urban wastewater systems. *Environmental Modelling and Software*, 20(4 SPEC. ISS.):415–426.

Cappai, G., Carucci, A., and Onnis, A. (2004). Use of industrial wastewaters for the optimization and control on nitrogen removal processes. *Water Science and Technology*, 50(6):17–24.

Carucci, A., Rolle, E., and Smurra, P. (1999). Management optimisation of a large wastewater treatment plant. *Water Science and Technology*, 39(4):129–136.

Cavin, L., Fischer, U., Mŏsat́, A., and Hungerbühler, K. (2005). Batch process optimization in a multipurpose plant using tabu search with a design-space diversification. *Computers and Chemical Engineering*, 29(8):1770–1786.

Chachuat, B., Roche, N., and Latifi, M. A. (2001). Dynamic optimisation of small size wastewater treatment plants including nitrification and denitrification processes. *Computers and Chemical Engineering*, 25(4-6):585–593.

Chachuat, B., Singer, A. B., and Barton, P. I. (2006). Global methods for dynamic optimization and mixed-integer dynamic optimization. *Industrial and Engineering Chemistry Research*, 45(25):8373–8392.

Chelouah, R. and Siarry, P. (2003). Genetic and nelder-mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. *European Journal of Operational Research*, 148(2):335–348.

Chen, C. T. and Hwang, C. (1990a). Optimal control computation for differential-algebraic process systems with general constraints. *Chemical Engineering Communications*, 97:9–26.

Chen, C. T. and Hwang, C. (1990b). Optimal on-off control for fed-batch fermentation processes. *Industrial and Engineering Chemistry Research*, 29(9):1869–1875.

Chen, D.-S. D., Singh, R. K., Haghighi, K., and Nelson, P. E. (1993). Finite element analysis of temperature distribution in microwaved cylindrical potato tissue. *Journal of Food Engineering*, 18(4):351–368.

Chiou, J.-P. and Wang, F.-S. (1999). Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process. *Computers and Chemical Engineering*, 23(9):1277–1291.

Chunfeng, W. and Xin, Z. (2002). Ants foraging mechanism in the design of multiproduct batch chemical process. *Industrial and Engineering Chemistry Research*, 41(26):6678–6686.

Coello Coello, C. A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287.

Coleman, T. F. and Li, Y. (1994). On the convergence of interior-reflective newton methods for nonlinear minimization subject to bounds. *Mathematical Programming*, 67(1-3):189–224.

Copp, J. (2001). *The COST Simulation Benchmark - Description and Simulator Manual*. COST (European Cooperation in the field of Scientific and Technical Research), Brussels.

Costa, L. and Oliveira, P. (2001). Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. *Computers and Chemical Engineering*, 25(2-3):257–266.

Cox, D. D. and John, S. (1997). SDO: A statistical method for global optimization. In Alexandrov, N. and Hussaini, M., editors, *Multidisciplinary Design Optimization: State of the Art*, pages 315–329. SIAM, Philadelphia, PA.

Csendes, T. (1988). Nonlinear parameter estimation by global optimization - efficiency and reliability. *Acta Cybernetica*, 8(4):361–370.

Cuthrell, J. E. and Biegler, L. T. (1989). Simultaneous optimization and solution methods for batch reactor control profiles. *Computers and Chemical Engineering*, 13(1-2):49–62.

Davis, E. and Ierapetritou, M. (2007). A kriging method for the solution of nonlinear programs with black-box functions. *AIChE Journal*, 53(8):2001–2012.

de la Maza, M. and Yuret, D. (1994). Dynamic hill climbing. *AI Expert*, 9(3):26–31.

Dennis, J. E., Gay, D. M., and Welsch, R. E. (1981). An adaptive non-linear least-squares algorithm. *ACM Transactions on Mathematical Software*, 7(3):348–368.

Dennis, J. E. J. (1977). Nonlinear least squares and equations. In Jacobs, D., editor, *State of the Art in Numerical Analysis*, pages 269–312. Academic Press.

Dietrich, C. R. and Osborne, M. R. (1991). Estimation of covariance parameters in kriging via restricted maximum likelihood. *Mathematical Geology*, 23(1):119–135.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming, Series B*, 91(2):201–213.

Dolan, E. D., Moré, J. J., and Munson, T. S. (2004). Benchmarking optimization problems with cops 3.0. Technical Report ANL/MCS-TM-273, Argonne National Laboratory.

Dorigo, M. (1992). *Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale*. PhD thesis, Politecnico di Milano, Italy.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. MIT Press, Cambridge, MA.

Dréo, J., Aumasson, J. P., Tfaili, W., and Siarry, P. (2007). Adaptive learning search, a new tool to help comprehending metaheuristics. *International Journal on Artificial Intelligence Tools*, 16(3):483–505.

Dréo, J., Petrowski, A., Taillard, E., and Siarry, P. (2006). *Metaheuristics for Hard Optimization Methods and Case Studies*. Springer.

Dréo, J. and Siarry, P. (2006). An ant colony algorithm aimed at dynamic continuous optimization. *Applied Mathematics and Computation*, 181(1):457–467.

Duran, M. A. and Grossmann, I. E. (1986). Outer-approximatino algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36(3):307–339.

Eberhart, R. C., Shi, Y., and Kennedy, J. (2001). *Swarm Intelligence*. Morgan Kaufmann.

Egea, J. A., Rodríguez-Fernández, M., Banga, J. R., and Martí, R. (2007a). Scatter search for chemical and bio-process optimization. *Journal of Global Optimization*, 37(3):481–503.

Egea, J. A., Vazquez, E., Banga, J. R., and Martí, R. (2007b). Improved scatter search for the global optimization of computationally expensive dynamic models. *Journal of Global Optimization*, In press(doi:10.1007/s10898-007-9172-y).

Egea, J. A., Vries, D., Alonso, A. A., and Banga, J. R. (2007c). Global optimization for integrated design and control of computationally expensive process models. *Industrial and Engineering Chemistry Research*, 46:9148–9157.

Esposito, W. R. and Floudas, C. A. (2000a). Deterministic global optimization in nonlinear optimal control problems. *Journal of Global Optimization*, 17(1-4):97–126.

Esposito, W. R. and Floudas, C. A. (2000b). Global optimization for the parameter estimation of differential-algebraic systems. *Industrial and Engineering Chemistry Research*, 39(5):1291–1310.

Exler, O., Antelo, L. T., Egea, J. A., Alonso, A. A., and Banga, J. R. (2008). A tabu search-based algorithm for mixed-integer nonlinear problems and its application to integrated process and control system design. *Computers and Chemical Engineering*, 32(8):1877–1891.

Exler, O. and Schittkowski, K. (2006). MISQP: A fortran implementation of a trust region SQP algorithm for mixed-integer nonlinear programming - user's guide, version 2.1-. Technical report, Department of Computer Science University of Bayreuth, Bayreuth, Germany.

Exler, O. and Schittkowski, K. (2007). A trust region SQP algorithm for mixed-integer nonlinear programming. *Optimization Letters*, 1(3):269–280.

Faber, R., Jockenhövel, T., and Tsatsaronis, G. (2005). Dynamic optimization with simulated annealing. *Computers and Chemical Engineering*, 29(2):273–290.

Feo, T. A. and Resende, M. G. C. (1989). Probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67–71.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133.

Fleurent, C., Glover, F., Michelon, P., and Valli, Z. (1996). Scatter search approach for unconstrained continuous optimization. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pages 643–648.

Floudas, C. A. (2000). *Deterministic Global Optimization: Theory, Methods and Applications*. Kluwer Academics, The Netherlands.

Floudas, C. A., Akrotirianakis, I. G., Caratzoulas, S., Meyer, C. A., and Kallrath, J. (2005). Global optimization in the 21st century: Advances and challenges. *Computers and Chemical Engineering*, 29(6):1185–1202.

Floudas, C. A. and Pardalos, P. M. (2000). Recent developments in deterministic global optimization and their relevance to process design. *AICHE Symposium Series*, 96(323):84–98.

Fraga, E. S. (2006). Hybrid methods for optimisation. In Zilinskas, J. and Bogle, I. D. L., editors, *Computer Aided Methods for Optimal Design and Operations*, pages 1–14. World Scientific, New Jersey.

Fraga, E. S. and Senos Matias, T. R. (1996). Synthesis and optimization of a nonideal distillation system using a parallel genetic algorithm. *Computers and Chemical Engineering*, 20(SUPPL.1):S79–S84.

Fraga, E. S. and Žilinskas, A. (2003). Evaluation of hybrid optimization methods for the optimal design of heat integrated distillation sequences. *Advances in Engineering Software*, 34(2):73–86.

Fu, G., Butler, D., and Khu, S. T. (2008). Multiple objective optimal control of integrated urban wastewater systems. *Environmental Modelling and Software*, 23(2):225–234.

Fuguitt, R. E. and Hawkins, J. E. (1947). Rate of thermal isomerization of α-pinene in the liquid phase. *J. A. C. S.*, 69:461.

García, M. S. G., Balsa-Canto, E., Alonso, A. A., and Banga, J. R. (2006). Computing optimal operating policies for the food industry. *Journal of Food Engineering*, 74(1):13–23.

Garrard, A. and Fraga, E. S. (1998). Mass exchange network synthesis using genetic algorithms. *Computers and Chemical Engineering*, 22(12):1837–1850.

Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H. (1989). Constrained nonlinear programming. In Nemhauser, G. L., Rinnooy Kan, A. H. G., and Todd, M. J., editors, *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*, pages 171–210. Elsevier North-Holland, Inc., New York, NY, USA.

Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H. (1998). User's guide for npsol 5.0: A fortran package for nonlinear programming. Technical Report SOL 86-1, Systems Optimization Laboratory, Stanford University.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8:156–166.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549.

Glover, F. (1989). Tabu search part i. *ORSA Journal on Computing*, 1(3):190–206.

Glover, F. (1990). Tabu search part ii. *ORSA Journal on Computing*, 2(1):4–32.

Glover, F. (1994). Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Applied Mathematics*, 49(1-3):231–255.

Glover, F. (1998). A template for scatter search and path relinking. In Hao, J. K., Lutton, E., Ronald, E., Schoenauer, M., and Snyers, D., editors, *Artificial Evolution*, volume 1363 of *Lecture Notes in Computer Science*, pages 13–54. Springer Verlag.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers, Norwell, MA.

Glover, F., Laguna, M., and Martí, R. (2000a). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3):652–684.

Glover, F., Laguna, M., and Martí, R. (2003). Scatter search. In Ghosh, A. and Tsutsui, S., editors, *Advances in Evolutionary Computation: Theory and Applications*, pages 519–537. Springer-Verlag, New York.

Glover, F., Laguna, M., and Martí, R. (2007). Principles of tabu search. In Gonzalez, T., editor, *Approximation Algorithms and Metaheuristics*, volume 23, pages 1–12. Chapman & Hall/CRC.

Glover, F., Løkketangen, A., and Woodruff, D. L. (2000b). Scatter search to generate diverse MIP solutions. In Laguna, M. and González Velarde, J. L., editors, *Computing tools for modeling optimization and simulation*, pages 299–320. Kluwer Academic Publishers, Boston.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison Wesley Longman.

Gras, R., Hernandez, D., Hernandez, P., Zangge, N., Mescam, Y., Frey, J., Martin, O., Nicolas, J., and Appel, R. D. (2003). Cooperative metaheuristics for exploring proteomic data. *Artificial Intelligence Review*, 20(1):95–120.

Groep, M. E., Gregory, M. E., Kershenbaum, L. S., and Bogle, I. D. L. (2000). Performance modeling and simulation of biochemical process sequences with interacting unit operations. *Biotechnology and Bioengineering*, 67(3):300–311.

Grossmann, I. E. (1996). *Global Optimization in Engineering Design*. Kluwer Academic Publishers.

Gutiérrez, G. and Vega, P. (2000). Integrated design of activated sludge process taking into account the closed loop controllability. In *Proceedings of ESCAPE-10*, pages 63–69, Firenze.

Gutmann, H.-M. (2001). A radial basis function method for global optimization. *Journal of Global Optimization*, 19(3):201–227.

Guus, C., Boender, E., and Romeijn, H. E. (1995). Stochastic methods. In Horst, R. and Pardalos, P. M., editors, *Handbook of Global Optimization*, pages 829–869. Kluwer Academic Publishers.

Hansen, N., Müller, S. D., and Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18.

Harjunkoski, I., Westerlund, T., Pörn, R., and Skrifvars, H. (1998). Different transformations for solving non-convex trim-loss problems by MINLP. *European Journal of Operational Research*, 105(3):594–603.

Hedar, A. (2004). *Studies on Metaheuristics for Continuous Global Optimization Problems*. PhD thesis, Kyoto University, Japan.

Henze, M., GradyJr, C. P. L., Gujer, W., Marais, G. V. R., and T., M. (1987). Activated sludge model n 1. IAWQ Scientific and Technical Report 1, IAWQ, London (Great Britain).

Herrera, F. and Lozano, M. (2000). Gradual distributed real-coded genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):43–62.

Herrera, F., Lozano, M., and Molina, D. (2006). Continuous scatter search: An analysis of the integration of some combination methods and improvement strategies. *European Journal of Operational Research*, 169(2):450–476.

Hindmarsh, A. C. (1980). LSODE and LSODI, two new initial value ordinary differential equation solvers. *ACM-SIGNUM Newsletter*, 15(4):10–11.

Hindmarsh, A. C. (1983). ODEPACK, a systematized collection of ode solvers. In Stepleman, R., editor, *Scientific Computing*. North Holland, Amsterdam, The Netherlands. www.netlib.org/odepack/lsode.f.

Hirsch, M. J., Meneses, C. N., Pardalos, P. M., and Resende, M. G. C. (2007). Global optimization by continuous grasp. *Optimization Letters*, 1(2):201–212.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* The University of Michigan Press.

Holmström, K. (2007). An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization. *Journal of Global Optimization*, In press(doi:10.1007/s10898-007-9256-8).

Holmström, K. and Edvall, M. M. (2004). The tomlab optimization environment. In Kallrath, J., editor, *Modeling Languages in Mathematical Optimization*, volume 88 of *Applied Optimization*, chapter 19, pages 369–378. Kluwer Academic Publishers, Boston, MA.

Hong, J. (1986). Optimal substrate feeding policy for a fed batch fermentation with substrate and product inhibition kinetics. *Biotechnology and Bioengineering*, 28(9):1421–1431.

Huang, D., Allen, T. T., Notz, W. I., and Zeng, N. (2006). Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34(3):441–466.

Hunter, W. G. and McGregor, J. F. (1967). The estimation of common parameters from several responses: Some actual examples. Technical report, The Department of Statistics, University of Winsconsin.

Jayaraman, V. K., Kulkarni, B. D., Karale, S., and Shelokar, P. (2000). Ant colony framework for optimal design and scheduling of batch plants. *Computers and Chemical Engineering*, 24(8):1901–1912.

Jeppsson, U. (1996). *Modelling Aspects of Wastewater Treatment Processes.* PhD thesis, Industrial Electrical Engineering and Automatics (IEA), Lund Institute of Technology (LTH).

Jeppsson, U. and Pons, M. N. (2004). The cost benchmark simulation model-current state and future perspective. *Control Engineering Practice*, 12(3):299–304.

Jiang, S., Ziver, A. K., Carter, J. N., Pain, C. C., Goddard, A. J. H., Franklin, S., and Phillips, H. J. (2006). Estimation of distribution algorithms for nuclear reactor fuel management optimisation. *Annals of Nuclear Energy*, 33(11-12):1039–1057.

Jin, R., Chen, W., and Simpson, T. W. (2001). Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization*, 23(1):1–13.

Jones, D. R. (2001a). DIRECT global optimization algorithm. In Floudas, C. A. and Pardalos, P. M., editors, *Encyclopedia of optimization*, pages 431–440. Kluwer Academic Publishers, Dordrecht.

Jones, D. R. (2001b). A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492.

Kallrath, J. (2000). Mixed integer optimization in the chemical process industry: Experience, potential and future perspectives. *Chemical Engineering Research and Design*, 78(6):809–822.

Kallrath, J. (2005). Solving planning and design problems in the process industry using mixed integer and global optimization. *Annals of Operations Research*, 140(1):339–373.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942–1948.

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

Kocis, G. R. and Grossmann, I. E. (1988). Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis. *Industrial and Engineering Chemistry Research*, 27(8):1407–1421.

Kookos, I. K. (2004). Optimization of batch and fed-batch bioreactors using simulated annealing. *Biotechnology Progress*, 20(4):1285–1288.

Krige, D. G. (1951). A statistical approach to some mine valuations and allied problems at the witwatersrand. Master's thesis, University of Witwatersrand.

Kuzmic, P. (1996). Program DYNAFIT for the analysis of enzyme kinetic data: application to hiv proteinase. *Analytical Biochemistry*, 237:260–273.

Ladiges, G. and Günner, C. (2003). Theoretical and practical results of the optimisation of hamburg's wwtps with dynamic simulation. *Water Science and Technology*, 47(12):27–33.

Ladiges, G., Guünner, C., and Otterpohl, R. (1999). Optimisation of the hamburg wastewater treatment plants by dynamic simulation. *Water Science and Technology*, 39(4):37–44.

Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E. (1999). Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147.

Laguna, M. and Martí, R. (2002). The OptQuest callable library. In Voss, S. and Woodruff, D. L., editors, *Optimization Software Class Libraries*. Kluwer Academic Publishers, Boston.

Laguna, M. and Martí, R. (2003). *Scatter Search: Methodology and Implementations in C*. Kluwer Academic Publishers, Boston.

Laguna, M. and Martí, R. (2005). Experimental testing of advanced scatter search designs for global optimization of multimodal functions. *Journal of Global Optimization*, 33(2):235–255.

Landa Becerra, R. and Coello-Coello, C. A. (2006). Cultured differential evolution for constrained optimization. *Computer Methods in Applied Mechanics and Engineering*, 195(33-36):4303–4322.

Larrañaga, P. and Lozano, J. A. (2001). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*.

Lasdon, L. and Plummer, J. C. (2008). Multistart algorithms for seeking feasibility. *Computers and Operations Research*, 35(5):1379–1393.

Leyffer, S. (2001). Integrating SQP and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization and Applications*, 18(3):295.

Lin, B. and Miller, D. (2004). Tabu search algorithm for chemical process optimization. *Computers and Chemical Engineering*, 28(11):2287–2306.

Lin, Y. and Stadtherr, M. A. (2006). Deterministic global optimization for parameter estimation of dynamic systems. *Industrial and Engineering Chemistry Research*, 45(25):8438–8448.

Lin, Y. E., Anantheswaran, R. C., and Puri, V. M. (1995). Finite element analysis of microwave heating of solid foods. *Journal of Food Engineering*, 25(1):85–112.

Lozano, M., Herrera, F., Krasnogor, N., and Molina, D. (2004). Real-coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation*, 12(3):273–302.

Luus, R. (1993a). Application of dynamic programming to differential-algebraic process systems. *Computers and Chemical Engineering*, 17(4):373–377.

Luus, R. (1993b). Piecewise linear continuous optimal control by iterative dynamic programming. *Industrial and Engineering Chemistry Research*, 32(5):859–865.

Luyben, K. C. A. M., Liou, J. K., and Bruin, S. (1982). Enzyme degradation during drying. *Biotechnology and Bioengineering*, 24(3):533–552.

Martí, R. (2003). Multi-start methods. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, pages 355–368. Kluwer Academic Publishers.

Martí, R. and Laguna, M. (2003). Scatter search: Diseño básico y estrategias avanzadas. *Revista Iberoamericana de Inteligencia Artificial*, 19:123–130.

Martí, R., Laguna, M., and Glover, F. (2006). Principles of scatter search. *European Journal of Operational Research*, 169(2):359–372.

Martin, J. D. and Simpson, T. W. (2005). Use of kriging models to approximate deterministic computer models. *AIAA Journal*, 43(4):853–863.

Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, 58:1246–1266.

Matyas, J. (1965). Random optimization. *Automation Remote Control*, 26:246–253.

Melián, B., Moreno Pérez, J. A., and Moreno Vega, J. M. (2003). Metaheurísticas: una visión global. *Revista Iberoamericana de Inteligencia Artificial*, 19:7–28.

Mendes, P. and Kell, D. B. (1998). Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *Bioinformatics*, 14 (10):869–883.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer.

Michalewicz, Z. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32.

Michalewicz, Z., Logan, T., and Swaminathan, S. (1994). Evolutionary operators for continuous convex parameter spaces. *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 84–97.

Michalewicz, Z. and Siarry, P. (2008). Feature cluster on adaptation of discrete metaheuristics to continuous optimization. *European Journal of Operational Research*, 185(3):1060–1061.

Michna, M. (2000). Model of the winding factor of the electrical machines. Technical report, Polytechnika Gdanska.

Mishkin, M., Karel, M., and Saguy, I. (1982). Applications of optimization in food dehydration. *Food Technology*, 36(7):101–109.

Moles, C. G., Gutierrez, G., Alonso, A. A., and Banga, J. R. (2003a). Integrated process design and control via global optimization: a wastewater treatment plant case study. *Chemical Engineering Research and Design*, 81:507–517.

Moles, C. G., Mendes, P., and Banga, J. R. (2003b). Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Research*, 13:2467–2474.

Molina, D., Herrera, F., and Lozano, M. (2005). Adaptive local search parameters for real-coded memetic algorithms. In *IEEE Congress on Evolutionary Computation, IEEE CEC 2005*, volume 1, pages 888–895.

Moré, J. J. (1978). The levenberg-marquardt algorithm: implementation and theory. In Watson, G. A., editor, *Numerical Analysis*, number 630 in Lecture Notes in Mathematics, pages 105–116. Springer-Verlag, Berlin.

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. C3P Report 826, Caltech Concurrent Computation Program.

Mühlenbein, H. and Paass, G. (1996). From recombination of genes to the estimation of distributions i. binary parameters. In *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*.

Najafpour, G. (2006). *Biochemical Engineering and Biotechnology*. Elsevier Science, The Netherlands.

Neumaier, A., Shcherbina, O., Huyer, W., and Vinkó, T. (2005). A comparison of complete global optimization solvers. *Mathematical Programming*, 103(2):335–356.

Ohlsson, T. and Bengtsson, N. E. (1971). Microwave heating profiles in food. *Microwave Energy Applications Newsletter*, 4(6):1–8.

Osman, I. H. and Kelly, J. P. (1996). *Meta-heuristics: Theory and applications*. Kluwer Academic Publishers, Boston.

Osman, I. H. and Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, 63:513623.

Ourique, C. O., Biscaia Jr., E. C., and Pinto, J. C. (2002). The use of particle swarm optimization for dynamical analysis in chemical processes. *Computers and Chemical Engineering*, 26(12):1783–1793.

Panier, E. and Tits, A. L. (1993). On combining feasibility, descent and superlinear convergence in inequality constrained optimization. *Mathematical Programming*, 59:261–276.

Papamichail, I. and Adjiman, C. S. (2002). A rigorous global optimization algorithm for problems with ordinary differential equations. *Journal of Global Optimization*, 24:1–33.

Pardalos, P. M., Romeijn, H. E., and Tuy, H. (2000). Recent developments and trends in global optimization. *Journal of Computational and Applied Mathematics*, 124(1-2):209–228.

Park, C. and Lee, T.-Y. (2004). Optimal control by evolutionary algorithm technique combined with spline approximation method. *Chemical Engineering Communications*, 191(2):262–277.

Pinter, J. (1996). *Global Optimization in Action. Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications*. Kluwer Academics, Netherlands.

Polisetty, P. K., Voit, E. O., and Gatzke, E. P. (2006). Identification of metabolic system parameters using global optimization methods. *Theoretical Biology and Medical Modelling*, 3:4.

Ponsich, A., Azzaro-Pantel, C., Domenech, S., and Pibouleau, L. (2007). Some guidelines for genetic algorithm implementation in MINLP batch plant design problems. In Siarry, P. and Michalewicz, Z., editors, *Advances in Metaheuristics for Hard Optimization*, Natural Computing Series, pages 293–315. Springer.

Preux, P. and Talbi, E.-G. (1999). Towards hybrid evolutionary algorithms. *International Transactions in Operational Research*, 6:557–570.

Price, K. V., Storn, R. M., and Lampinen, J. A. (2005). *Differential Evolution A Practical Approach to Global Optimization*. Natural Computing Series. Springer-Verlag, Berlin, Germany.

Price, W. L. (1983). Global optimization by controlled random search. *Journal of Optimization Theory and Applications*, 40(3):333348.

Rajesh, J., Gupta, K., Kusumakar, H. S., Jayaraman, V. K., and Kulkarni, B. D. (2001). Dynamic optimization of chemical processes using ant colony framework. *Computers and Chemistry*, 25(6):583–595.

Rajesh, J., Gupta, K., Kusumakar, H. S., Jayaraman, V. K., and Kulkarni, B. D. (2003). A tabu search based approach for solving a class of bilevel programming problems in chemical engineering. *Journal of Heuristics*, 9(4).

Rastrigin, L. A. and Rubinstein, Y. (1969). The comparison of random search and stochastic approximation while solving the problem of optimization. *Automatic Control*, 2:2329.

Rechenberg, I. (1973). *Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog.

Regis, R. G. and Shoemaker, C. A. (2007a). Improved strategies for radial basis function methods for global optimization. *Journal of Global Optimization*, 37(1):113–135.

Regis, R. G. and Shoemaker, C. A. (2007b). Parallel radial basis function methods for the global optimization of expensive functions. *European Journal of Operational Research*, 127(2):514–535.

Rich, E. and Knight, K. (1991). *Artificial Intelligence*. McGraw-Hill.

Rigopoulos, S. and Linke, P. (2002). Systematic development of optimal activated sludge process designs. *Computers and Chemical Engineering*, 26(4-5):585–597.

Rinnooy-Kan, A. H. G. and Timmer, G. T. (1987). Stochastic global optimization methods. Part I: Clustering methods. *Mathematical Programming*, 39:27.

Rodríguez-Acosta, F., Regalado, C., and Torres, N. (1999). Non-linear optimization of biotechnological processes by stochastic algorithms. Application to the maximization of the production rate of ethanol, glycerol and carbohydrates by saccharomyces cerevisiae. *Journal of Biotechnology*, 68(1):15–28.

Rodríguez-Fernández, M. (2006). *Modelado e Identificación de Bioprocesos*. PhD thesis, University of Vigo, Spain.

Rodríguez-Fernández, M., Balsa-Canto, E., Egea, J. A., and Banga, J. R. (2007). Identifiability and robust parameter estimation in food process modeling: Application to a drying model. *Journal of Food Engineering*, 83(3):374–383.

Rodríguez-Fernández, M., Egea, J. A., and Banga, J. R. (2006a). Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. *BMC Bioinformatics*, 7:483+.

Rodríguez-Fernández, M., Mendes, P., and Banga, J. R. (2006b). A hybrid approach for efficient and robust parameter estimation in biochemical pathways. *BioSystems*, 83:248–265.

Roubos, J. A., Van Straten, G., and Van Boxtel, A. J. B. (1999). An evolutionary strategy for fed-batch bioreactor optimization: concepts and performance. *Journal of Biotechnology*, 67(2-3):173–187.

Runarsson, T. P. and Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4:284–294.

Saa, J., Alonso, A. A., and Banga, J. R. (1998). Optimal control of microwave heating using mathematical models of medium complexity. In *Automatic Control of Food and Biological Processes (AcoFop IV)*, Göteborg (Sweden).

Sakizlis, V., Perkins, J. D., and Pistikopoulos, E. N. (2004). Recent advances in optimization-based simultaneous process and control design. *Computers and Chemical Engineering*, 28(10):2069–2086.

Sandgren, E. (1990). Nonlinear integer and discrete programming in mechnical design optimization. *Journal of Mechanisms, Transmissions, and Automation in Design*, 112(2):223–229.

Sarkar, D. and Modak, J. M. (2004). Optimization of fed-batch bioreactors using genetic algorithm: Multiple control variables. *Computers and Chemical Engineering*, 28(5):789–798.

Sasena, M., Papalambros, P., and Goovaerts, P. (2002). Exploration of metamodeling sampling criteria for constrained global optimization. *Engineering Optimization*, 34:263–278.

Sastry, K. and Goldberg, D. (2005). Genetic algorithms. In Burke, E. K. and Kendall, G., editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 97–125. Springer, New York, NY, USA.

Schiesser, W. E. (1991). *The numerical method of lines*. Academic Press, New York, USA.

Schittkowski, K. (2002). *Numerical data fitting in dynamical systems - A practical introduction with applications and software*. Kluwer Academic Publishers.

Schütze., Butler, D., and Beck, M. B. (1999). Optimisation of control strategies for the urban wastewater system - an integrated approach. *Water Science and Technology*, 39(9):209–216.

Schütze, M., Butler, D., and Beck, M. B. (2001). Parameter optimisation of real-time control strategies for urban wastewater systems. *Water Science and Technology*, 43(7):139–146.

Schütze, M., Butler, D., Beck, M. B., and Verworn, H. . (2002). Criteria for assessment of the operational potential of the urban wastewater system. *Water Science and Technology*, 45(3):141–148.

Schweiger, C. A. and Floudas, C. A. (1997). Interaction of design and control: Optimization with dynamic models. In Hager, W. W. and Pardalos, P. M., editors, *Optimal Control: Theory, Algorithms, and Applications*, pages 388–435. Kluwer Academic Publishers.

Sendín, O. H., Moles, C. G., Alonso, A. A., and Banga, J. R. (2004). Multi-objective integrated design and control using stochastic global optimization methods. In Seferlis, P. and Georgiadis, M. C., editors, *The integration of process design and control*, pages 555–581. Elsevier, Amsterdam-Boston.

Shampine, L. F. and Watts, H. A. (1977). The art of writing a runge-kutta code, part 1. In Rice, J. R., editor, *Mathematical Software III*, pages 257–275. Academic Press, New York.

Shelokar, P. S., Jayaraman, V. K., and Kulkarni, B. D. (2008). Multicanonical jump walk annealing assisted by tabu for dynamic optimization of chemical engineering processes. *European Journal of Operational Research*, 185(3):1213–1229.

Shimizu, K. (1996). A tutorial review on bioprocess systems engineering. *Computers and Chemical Engineering*, 20(6-7):915–941.

Shuler, M. L. and Kargi, F. (1992). *Bioprocess Engineering*. Prentice Hall, Englewood Cliffs, New Jersey.

Simpson, T. W., Peplinski, J. D., Koch, P. N., and Allen, J. K. (2001). Metamodels for computer-based engineering design: Survey and recommendations. *Engineering with Computers*, 17(2):129–150.

Singer, A. B., Bok, J. K., and Barton, P. I. (2001). Convex underestimators for variational and optimal control problems. *Computer Aided Chem. Eng.*, 9:767–772.

Socha, K. and Dorigo, M. (2008). Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185(3):1155–1173.

Srinivas, M. and Rangaiah, G. P. (2007). Differential evolution with tabu list for global optimization and its application to phase equilibrium and parameter estimation problems. *Industrial and Engineering Chemistry Research*, 46(10):3410–3421.

Steffens, M. A., Fraga, E. S., and Bogle, I. D. L. (2000). Synthesis of bioprocesses using physical properties data. *Biotechnology and Bioengineering*, 68(2):218–230.

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer, New York.

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

Sun, D. Y. and Lin, P. M. (2006). The solution of time optimal control problems by simulated annealing. *Journal of Chemical Engineering of Japan*, 39(7):753–766.

Taillard, E. D., Gambardella, L. M., Gendreau, M., and Potvin, J.-Y. (2001). Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operational Research*, 135(1):1–16.

Takács, I., Patry, G. G., and Nolasco, D. (1991). A dynamic model of the clarification-thickening process. *Water Research*, 25(10):1263–1271.

Talbi, E.-G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564.

Teh, Y. S. and Rangaiah, G. P. (2003). Tabu search for global optimization of continuous functions with application to phase equilibrium calculations. *Computers and Chemical Engineering*, 27(11):1665–1679.

Tfaili, W. and Siarry, P. (2008). A new charged ant colony algorithm for continuous dynamic optimization. *Applied Mathematics and Computation*, 197(2):604–613.

Tjoa, I. B. and Biegler, L. T. (1991). Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. *Industrial and Engineering Chemistry Research*, 30(2):376–385.

Törn, A. A. (1973). Global optimization as a combination of global and local search. In *Proceedings of Computer Simulation Versus Analytical Solutions for Business and Economic Models*.

Trafalis, T. B. and Kasap, S. (1996). An affine scaling scatter search approach for continuous global optimization problems. In Dagli, C. H., Akay, M., Chen, C. L. P., Fernandez, B. R., and Ghosh, J., editors, *Intelligent Engineering Systems Through Artificial Neural Networks*, volume 6, pages 1027–1032. ASME Press.

Trafalis, T. B. and Kasap, S. (2002). A novel metaheuristics approach for continuous global optimization. *Journal of Global Optimization*, 23(2):171–190.

Tsai, K. and Wang, F. (2005). Evolutionary optimization with data collocation for reverse engineering of biological networks. *Bioinformatics*, 21(7):1180–1188.

Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., and Martí, R. (2005). A multistart scatter search heuristic for smooth NLP and MINLP problems. In Rego, C. and Alidaee,

B., editors, *Adaptive Memory and Evolution: Tabu Search and Scatter Search*, pages 25–58. Kluwer Academic Publishers.

Vanrolleghem, P. A. and Dochain, D. (1998). Bioprocess model identification. In Van Impe, J. F., Vanrolleghem, P. A., and Iserentant, D. M., editors, *Advanced Instrumentation, data interpretation, and control of biotechnological process*, pages 251–318. Kluwer Academic Publishers.

Vanrolleghem, P. A. and Gillot, S. (2002). Robustness and economic measures as control benchmark performance criteria. *Water Science and Technology*, 45(4-5):117–126.

Vassiliadis, V. S., Sargent, R. W. H., and Pantelides, C. C. (1994a). Solution of a class of multistage dynamic optimization problems. 1. problems without path constraints. *Industrial and Engineering Chemistry Research*, 33(9):2111–2122.

Vassiliadis, V. S., Sargent, R. W. H., and Pantelides, C. C. (1994b). Solution of a class of multistage dynamic optimization problems. 2. problems with path constraints. *Industrial and Engineering Chemistry Research*, 33(9):2123–2133.

Vazquez, E. (2005). *Modélisation comportementale de systèmes non-linéaires multivariables par méthodes á noyaux et applications*. PhD thesis, Paris XI Orsay University.

Vecchia, A. V. (1998). Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society*, B(50):297–312.

Villemonteix, J., Vazquez, E., and Walter, E. (2008). An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, Accepted:In Press.

Villota, R. and Karel, M. (1980a). Prediction of ascorbic acid retention during drying. i. moisture and temperature distribution in a model system. *Journal of Food Processing and Preservation*, 4:111–134.

Villota, R. and Karel, M. (1980b). Prediction of ascorbic acid retention during drying. ii. simulation of retention in a model system. *Journal of Food Processing and Preservation*, 4:141–159.

Vrugt, J. A. and Robinson, B. A. (2007). Improved evolutionary optimization from genetically adaptive multimethod search. *PNAS*, 104(3):708–711.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

Wan, X., Pekny, J. F., and Reklaitis, G. V. (2005). Simulation-based optimization with surrogate models: Application to supply chain management. *Computers and Chemical Engineering*, 29(6):1317–1328.

Wang, C., Quan, H., and Xu, X. (1999). Optimal design of multiproduct batch chemical processes using tabu search. *Computers and Chemical Engineering*, 23(3):427–437.

Wang, F.-S., Su, T.-L., and Jang, H.-J. (2001). Hybrid differential evolution for problems of kinetic parameter estimation and dynamic optimization of an ethanol fermentation process. *Industrial and Engineering Chemistry Research*, 40(13):2876–2885.

Wang, K., Salhi, A., and Fraga, E. S. (2004). Process design optimisation using embedded hybrid visualisation and data analysis techniques within a genetic algorithm optimisation framework. *Chemical Engineering and Processing*, 43(5):663–675.

Yaglom, A. M. (1987). *Correlation Theory of Stationary and Related Random Functions I: Basic results*. Springer Series in Statistics. Springer-Verlag, Inc., New York.

Ye, Y. (1987). *Interior algorithms for linear, quadratic and linearly constrained non-linear programming*. PhD thesis, Stanford University.

Yeniay, O. (2005). Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and Computational Applications*, 10(1):45–56.

Yuan, X., Zhang, S., Piboleau, L., and Domenech, S. (1988). Une methode d'optimization nonlineare en variables mixtes pour la conception de procedes. *RAIRO*, 22:31.

Zelinka, I., Vasek, V., Kolomaznik, K., and Dostal, P. (2001). Memetic algorithm and global optimization of chemical reactor. In *PC Control 2001, 13th International Conference on Process Control*, High Tatras, Slovakia.

Zhang, B., Chen, D., and Zhao, W. (2005). Iterative ant-colony algorithm and its application to dynamic optimization of chemical process. *Computers and Chemical Engineering*, 29(10):2078–2086.

Zhou, Z., Ong, Y. S., Lim, M. H., and Lee, B. S. (2007). Memetic algorithm using multi-surrogates for computationally expensive optimization problems. *Soft Computing*, 11(10):957–971.

Zwolak, J. W., Tyson, J. J., and Watson, L. T. (2005). Globally optimised parameters for a model of mitotic control in frog egg extracts. *IEE Proceedings Systems Biology*, 152(2):81–92.

Part VII

Publications

Publications

Publications in international journals

- Exler, O., Antelo, L. T., Egea, J. A., Alonso, A. A., and Banga, J. R. (2007). A tabu search-based algorithm for mixed-integer nonlinear problems and its application to integrated process and control system design. *Computers and Chemical Engineering*, In Press(doi:10.1016/j.compchemeng.2007.10.008).

- Egea, J. A., Vazquez, E., Banga, J. R., and Martí, R. (2007). Improved scatter search for the global optimization of computationally expensive dynamic models. *Journal of Global Optimization*, In press(doi:10.1007/s10898-007-9172-y).

- Egea, J. A., Vries, D., Alonso, A. A., and Banga, J. R. (2007). Global optimization for integrated design and control of computationally expensive process models. *Industrial and Engineering Chemistry Research*, 46:9148-9157.

- Rodríguez-Fernández, M., Balsa-Canto, E., Egea, J. A., and Banga, J. R. (2007). Identifiability and robust parameter estimation in food process modeling: Application to a drying model. *Journal of Food Engineering*, 83(3):374-383.

- Egea, J. A., Rodríguez-Fernández, M., Banga, J. R., and Martí, R. (2007). Scatter search for chemical and bio-process optimization. *Journal of Global Optimization*, 37(3):481-503.

- Rodríguez-Fernández, M., Egea, J. A., and Banga, J. R. (2006). Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. *BMC Bioinformatics*, 7:483+.

Book chapters

- Rodríguez-Fernández, M., Egea, J. A., and Banga, J. R. (2006). Novel metaheuristic for parameter estimation and optimal experimental design in Systems Biology. In: *Understanding and Exploiting Systems Biology in Bioprocesses and Biomedicine*, pp. 103-117. M. Cánovas, J. L. Iborra and A. Manjón (Eds.), Fundación Cajamurcia, Murcia (Spain).

Contributions to international congresses

- Egea, J. A., Vazquez, E., Banga, J. R., and Martí, R. (2007). Improved scatter search for the global optimization of computationally expensive dynamic models. Fifth International Conference on Advances in Global Optimization: Methods and Applications (AGO2007), July 13-17, 2007, Mykonos (Greece).

- Exler, O., Banga, J. R., Egea, J. A., Antelo, L. T. and Alonso, A. A. (2006). Metaheuristics for integration of process and control system design. 9th Conference on Process Integration, Modelling and Optimisation for Energy Saving and Pollution Reduction (PRESS 2006), August, 27-31, 2006, Prague (Czech Republic).

- Egea, J. A. (2006). An Optimization Approach for Integrated Design and Parameter Estimation in Process Engineering. 5th Conference of PhD Students in Computer Science, June 27-30, 2006, Szeged (Hungary).

- Rodríguez-Fernández, M., Egea, J. A., and Banga, J. R. (2006). Novel metaheuristic for parameter estimation and optimal experimental design in Systems Biology. 1st International Symposium on Systems Biology, June 1-2, 2006, Murcia, (Spain).

- Egea, J. A., Vries, D., Alonso, A. A., and Banga, J. R. (2005). Global Optimization for Integrated Design and Control of Computationally Expensive Process Models. European Control Conference (ECC 2005), December 12-15, 2005, Seville (Spain).

- Rodríguez-Fernández, M., Egea, J. A., and Banga, J. R. (2005). Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. European Conference on Mathematical and Theoretical Biology - ECMTB05, July 18-22, 2005, Dresden, (Germany).

Articles in preparation

- Schlüter, M., Egea, J. A., and Banga, J. R. (2008). Extended Ant Colony Optimization for non-convex Mixed Integer Nonlinear Programming. Submitted to *Computers and Operations Research*.

- Egea, J. A., García, M. S. G., Balsa-Canto, E., and Banga, J. R. (2008). Global Dynamic Optimization in Chemical Engineering using the Scatter Search Metaheuristic. To be submitted to *Industrial and Engineering Chemistry Research*.

- Schlüter, M., Antelo, L. T., Egea, J. A., Alonso, A. A., and Banga, J. R. (2008). An Ant Colony Optimization based algorithm for the integrated process and control system design. To be submitted to *Industrial and Engineering Chemistry Research*.