CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.

Pedro Jesús Copado méndez

Dipòsit Legal: T 1773-2014

Doctoral Thesis

Pedro Jesús Copado Méndez

CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS

Department of Chemical Engineering

University Rovira i Virgili

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

Pedro Jesús Copado Méndez

CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS

Doctoral Thesis

Supervised by:
Dr. Gonzalo Guillén
Dr. Laureano Jiménez

Department of Chemical Engineering
SUSCAPE

Tarragona
2014

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

UNIVERSITAT ROVIRA I VIRGILI

University Rovira i Virgili
Department of Chemical Engineering
Av. Països Catalans, 26
47007, Tarragona
Phone +34 977 55 86 43

Dr. Gonzalo Guillén, Dr. Laureano Jiménez

CERTIFY:

That the present study entitled "Contribution to the development of efficient algorithms for solving complex single-objective and multi-objective optimization models", presented by Pedro Jesús Copado Méndez for the award of the degree of Doctor, has been carried out under our supervision at the Chemical Engineering Department of the University Rovira i Virgili.

Tarragona, September 22, 2014

Dr. Gonzalo Guillén Dr. Laureano Jiménez

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX
SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

Acknowledgments

Quiero agradecer a mis supervisores Dr. Gonzalo Guillén Gosálbez y Dr. Laureano Jiménez Esteller por su supervisión, soporte y paciencia para la realización de este trabajo de investigación. Por otro lado, también quiero agradecer la colaboración prestada por el Dr. Christian Blum de la Universidad del Pais Vasco.

Quiero agradecer a todos mis compañeros y ex-compañeros del SUSCAPE, en espcial a Carmen María Torres, Núria Rovira, Carlos Pozo, Daniel Cortés y Robert Brunet, y a mis compañeros del DEQ, en especial a Sergio Daniel Rios, Sema Sirin y Kelly Briceño por los momentos tan divertidos que hemos pasado, por tanta tardes en la pp3/aula pont, por los cafés (sobretodo los cafés de los viernes), por todos los debates políticos, por colaborar como representante, por hacer el dichoso POA, por los asados, por los consejos consultivos, por las "queimadas", por las "cuchipandas", por las pedradas, por la isla del mojito, por su apoyo moral y por su amistad . . . inolvidables.

Por último me gustaría disculparme antes todos aquellos con los que también compartí momentos muy bonitos, pero no nombro: igualmente muchas gracias a vosotros también.

Summary

The aim of this thesis is to provide a decision-support tool for single/multi-optimization problems. The thesis is divided in several parts as follow.

- Firstly, a novel hybrid metaheuristic algorithm was developed in order to optimize single-objective (profit or cost) problems arising in the context of the strategic planning of supply chains (SCs). The strategic planning of SC consists of determining the number, location and capacities of the SC facilities to be established in each sub-region of a given country, their expansion policy over the planning horizon, the transportation links and number/type of transport that need to be established in the network, and the production rates and flows of the involved feedstocks, wastes, and final products, giving rise to optimal profit or cost. These problems are mathematically formulated as a mixed-integer linear programming (MILP) models which are solved by means of the proposed method. This algorithm consists of generating an initial solution, which is subsequently improved by the application of local search in a relative neighborhood of solutions. The construction of initial solutions should be fast (computationally not expensive), and – if possible – initial solutions should be a good starting point for local search. The large neighborhood search (LNS)

provides near optimal solutions in a fraction of time spent by CPLEX 12. The capabilities of the proposed approach are illustrated by solving two models based on the Argentinean Ethanol SC and Spanish Hydrogen SC, respectively. In order to encompass all possible conversion pathways, the solved models include production facilities depending on the utilized technology and, storage facilities depending on the used technology as well. The region of interest (i.e., Argentina or Spain) is subdivided into a number of sub-regions, where the SC facilities can be installed. Different types of transport are considered for carrying materials between the sub-regions. The complexity of this MILP model is mainly given by the number of integer and binary variables. The number of these variables increases with the number of time intervals and sub-regions. Thus, large-scale problems with long range planning horizons can be computationally intractable.

- Secondly, to develop a computational framework to reduce the dimensionality of multi-objective optimization (MOO) problems that identifies and eliminates in a systematic manner redundant criteria from the mathematical model. The method proposed builds a mixed-integer linear programming (MILP) formulation introduced in a previous work by the authors. MOO has recently gained wider interest in different domains of engineering and science, but the major limitation of this approach is that its complexity grows rapidly as we increase the number of objectives. The method proposed builds on a mixed-integer linear programming (MILP) formulation based on *divide and*

conquer paradigm. This new approach was assessed by its application to two problems related to biofuels. The first addresses the strategic planning of Spanish hydrogen SC, while the second deals with the MOO of metabolic networks. The first case study is retrieved from the Spanish hydrogen SC considered in the first part of this thesis, but we extended it in order to integrate the evaluation of the environmental sustainability through the Life Cycle Assessment (LCA) methodology. The second case deals the preferred enzymatic profiles that optimize the synthesis rate of a metabolite at minimum cost (minimum number of changes in these activities, i.e., minimum change in gene expression) and minimum increase in the concentration of intermediate metabolites in the fermentation of *Saccharomyces cerevisiae* for ethanol production.

- Thirdly, we enhanced the ϵ-method that is based on the combined use of a rigorous dimensionality reduction method with pseudo/quasi-random sequences. The improvements are relied on two main elements: (i) the use of rigorous objective reduction techniques that eliminate redundant objectives from the search, thereby simplifying the MOO problem from the viewpoints of generation and analysis of the Pareto solutions; and (ii) the application of pseudo/quasi-random sequences (i.e., uniform distribution an Sobol and Halton sequences) for generating the epsilon parameter values used in the single-objective auxiliary models solved during the iterations of the algorithm. We illustrate the capabilities of our approach through its application to two SC design problems in which we optimize the eco-

nomic performance or total cost along with a set of environmental metrics quantified following LCA principles. The case studies are taken from previous parts of this thesis. On the one hand, the optimal design of three-echelon hydrogen SC for vehicle use in Spain taking into account economic and environmental concerns. And on the other hand the optimal design of tree-echelon bioethanol network and associated planning decisions that maximize the net present value and minimize the environmental impact. The latter model, presented in part one, has been extended in order to integrate environmental metrics, including the individual categories considered in the Eco-indicator 99: damage to human health (HH), damage to eco-system quality (DTE), and damage to resources (DTR), along with the global warming potential (GWP) and the Eco-indicator99 (EI99).

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX
SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

Contents

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

List of Figures

List of Tables

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX
SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

Abbreviations

URV	**U**niversity **R**ovira i **V**irgili
ETSEQ	**E**scola **T**ècnica **S**uperior **D**'Enginyeria **Q**uímica
IP	**I**ntegrated **P**roject
Q1	**Q**uadrimester **1**st
Q2	**Q**uadrimester **2**nd

Dedicated to
my parents, without whom this work would not have been possible,
to to my uncle Antonio who died during the achievement of this work,
to my family and friends.

List of Algorithms

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX
SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

Part I

First Part: Introduction

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

Chapter 1

General objectives

The objectives of this thesis are:

- To devise a systematic framework for the single-optimization of spatially explicit models for supply chain design and planning.
- To propose and apply novel optimization frameworks based on multi-objective optimization (MOO), economic analysis, and environmental assessment tools.
- To develop a novel hybrid metaheuristic for the optimization of deterministic single objective problems in which all of the parameters are known in advance.
- To develop an effective dimensionality reduction framework for facilitating the solution procedure of multi-objective optimization problems (MOO).
- To enhance the ϵ-constraint method through the use of objective reduction techniques and random sequences.

Chapter 2

Introduction

This thesis introduces a set of advanced mathematical programming tools to assist decision-making in problems arising in process systems engineering (PSE), with particular emphasis on supply chain management and multi-objective optimization.

We start by reviewing the literature on mathematical models applied to supply chain management and we then review multi-objective optimization, with particular emphasis on its application to environmental problems that optimize a set of environmental impacts along with the economic performance of a system. We finally study the issue of how to reduce the computational burden of multi-objective optimization using dimensionality reduction methods and random sequences.

The remaining of this document is organized as follows: in chapter 3, we review mathematical and algorithmic concepts and describe the methods developed whereas in chapter 4 we present the results obtained, and finally, in chapter 5, the conclusions of the work are drawn and future research lines that could extend the framework proposed herein are outlined.

2.1 Mathematical models for supply chain management

A supply chain (SC) is a network of entities where input materials are transformed into finished products that are delivered to the end customer. The study of a SC as a whole is a relatively new discipline called supply chain management (SCM), that aims to integrate manufacturing plants with their suppliers and customers in an efficient manner (Shapiro, 2001). SCM has gained wider interest in both, academia and industry, given its potential to increase the benefits through an efficient coordination of the operations of supply, manufacturing and distribution carried out in a network. For a more detailed description, see section 6.2. SCM is aimed at a broader set of real-world applications, with particular emphasis on logistics and distribution, which usually involve linear models, that belong traditionally to the domain of operations research (Puigjaner and Espuña, 2006).

SCM problems can be classified into strategic, tactical and operational levels according to the temporal and spatial scales considered in the analysis (see section 6.2). In this thesis we will focus on the strategic level, which deals with decisions that have a long time effect, such as those related with the establishment of new facilities. In addition, we distinguish here between single-objective optimization problems (see section 6.2) and multi-objective optimization problems (MOO) (see sections 3.1.1.6 and 7.2).

SCM problems can be posed in mathematical terms as mixed-integer lineal programming (MILP) models (see Appendix A.1 and A.2), some of

which might be multi-objective (MOO) as well. See sections 6.2, 3.1.1.6 and 7.2 for more details. In this type of problems, we first formulate a superstructure of alternatives (see Figure C.1), that is, a mathematical representation that accounts for a set of options for establishing different production and storage facilities in a set of potential locations with known demand and prices. The algorithm must then identify the best alternatives considering a single (or several) objective functions. For more details see appendix A.

A general review on the application of mathematical programming techniques in SCM can be found in the work by Mula et al. (2010) whereas more specific reviews devoted to process industries have been presented by Grossmann (2005) and Papageorgiou (2009). Particularly, MILP is nowadays the most widely used modelling tool for solving strategic SCM problems. MILP formulations for SCM typically adopt fairly simple linear aggregated representations of capacity. Besides avoiding the numerical difficulties associated with dealing with nonlinearities, this simplification permits an easy adaptation to a wide range of industrial applications. In these MILPs, continuous variables represent materials flows and purchases and sales of products, whereas binary variables model tactical and/or strategic decisions related to the network configuration, such as selection of technologies, and establishment of facilities and transportation links (Guillén et al., 2006).

Recently, several spatially explicit SCM models based on MILP have appeared in the literature. Akgul et al. (2011) presented a spatially explicit MILP for the optimal design of a biothanol SC with the objective

of minimizing the total cost. The model seeks to optimize the locations of the bioethanol production plants, the biomass and bioethanol flows between regions, and the number of transport units required for transferring these products between several regions of Northern Italy. Dal-Mas et al. (2011) developed a dynamic spatially explicit MILP modeling framework devised to optimize the design and planning of biomass-based fuel SCs under uncertainty in market conditions considering different financial criteria. Mele et al. (2011) presented an MILP to optimize the design of SCs for the combined production of sugar and ethanol. In this work, the problem is formulated as a multi-objective MILP that optimizes simultaneously the economic performance of the network and several LCA metrics (Giarola et al., 2012a,b; Elghali et al., 2007).

Several strategies have been explored for the efficient solution of the aforementioned MILPs arising in SCM. These strategies can be roughly classified into two major groups: deterministic and stochastic approaches. The former ones provide a rigorous bound on the global optimal solution whereas the latter do not. This is typically accomplished at the expense of larger CPU times, which are required to ensure the quality of the solution found within the desired tolerance.

In addition, deterministic methods can rely on either solving the full space MILP using branch and cut techniques, or decomposing it into subproblems of smaller size between which an algorithm iterates until a termination criterion is satisfied. We should clarify that in many cases this second type of approaches make also use of branch and cut techniques, but

only for solving the sub-problems resulting from decomposing the original MILP and not for the solution of the original MILP itself.

Several decomposition strategies have been devised to exploit the underlying mathematical structure of the MILPs arising in SCM. Bok et al. (2000) developed a bi-level decomposition algorithm for an MILP model that maximizes the profit of a production-distribution network. This algorithm could halve the solution time compared to the rigorous branch and cut algorithm implemented in CPLEX. Guillén-Gosálbez et al. (2010) introduced a bi-level algorithm for solving the strategic planning of hydrogen SCs for vehicle use. This decomposition method achieved reductions of up to one order of magnitude in CPU time compared with the full space method (the whole model without decomposition, relaxation or approximations) while still providing near optimal solutions (i.e., with less than 1% of optimality gap).

Lagrangean decomposition has also been used in the context of strategic SCM problems. Gupta and Maranas (1999) applied Lagrangean decomposition to solve a planning problem that considered different products and manufacturing sites. The authors reported a solution with an optimality gap of 1.6%, reducing in one order of magnitude the CPU time required by CPLEX 4.0 to find a solution with a gap of 3.2%. You and Grossmann (2010) introduced a spatial decomposition algorithm based on the integration of Lagrangean relaxation and piecewise linear approximation to reduce the computational expense of solving multi-echelon SC design problems under uncertain customer demands. Chen and Pinto (2008) investigated the

application of various Lagrangean-based techniques to planning problems that allowed them to reduce considerably the computational burden method while still achieving optimality gaps of less than 2%.

Other solution methods applied to SCM problems have been Bender's decomposition (Geoffrion and Graves, 1974) and "rolling horizon" algorithms based on the original work by Jain and Palekar (2005). The former approach has been applied to strategic/tactical SCM problems with medium-large time horizons (Dogan and Goetschalckx, 1999; Mirhassani et al., 2000; Paquet et al., 2004; Soner Kara and Onut, 2010; Üster et al., 2007). In contrast, rolling horizon algorithms have been primarily used for solving operational SCM problems (Dimitriadis et al., 1997; Elkamel and Mohindra, 1999; Balasubramanian and Grossmann, 2004). In the recent past, this approach has also been adapted to deal with strategic SCM problems (Kostin et al., 2011b).

The difficulty in solving spatially explicit SCM problems (and more generally, any type of MILP) is highly dependent on the number of discrete variables, since they are responsible for the combinatorial complexity of the problem. This number of discrete variables increases with the number of time periods and sub-regions considered in the SC model. Models accounting for a large number of time periods and/or sub-regions may lead to branch-and-bound trees with a prohibitive number of nodes, thus making the MILP computationally intractable. To deal with this, we present in this thesis a hybrid metaheuristic (see section 3.1.2.8) that combines large neightbourhood (LNS) search with standard branch and cut techniques.

2.2 Challenges in multi-objective optimization

Many problems in PSE, including SCM problems, require the simultaneous optimization of more than one objective function. Multi-objective optimization (MOO) is widely used in many areas of science and engineering for simultaneously optimizing several objective functions subject to some equality and inequality constraints. MOO has been applied to many different fields (Claro J., 2012; Ustun O., 2012; Sharma A., 2012; Alonso M., 2012; Niknam T., Fard A. K., 2012), among others.

Several methods exist for tackling MOO problems, including ϵ-constraint method (Haimes, Y.Y.; Lasdon, L.S.; Wismer, 1971), the weighted-sum method (Zadeh, 1963), normal-boundary intersection (Das and Dennis, 1996) and goal programming (Charnes et al., 1955; Charnes and Cooper, 1961; Ijiri, 1965; Charnes et al., 1967). MOO has found many applications in process systems engineering, system biology. Among these publications we emphasize the optimal design of supply chains (Yue et al., 2014; Murthy Konda et al., 2011; Sabio et al., 2012; Guillén-Gosálbez et al., 2010; Lin et al., 2008; INGASON et al., 2008; QADRDAN et al., 2008), refineries (Gebreslassie et al., 2013; Zhang et al., 2014; Santibañez Aguilar et al., 2014; Murillo-alvarado and El-halwagi, 2013), and reverse osmosis networks (Du et al., 2014). We mention as well the work of Pozo and Guillén (2012) which is concerned with the optimization of metabolic pathways in system biology.

The solution to MOO problems is not usually a single solution but rather a large number of Pareto-optimal solutions. Clearly, testing all these alternatives in the industry would be prohibitive in terms of time and resources. Multi-criteria decision-making (MCDM) can be of great help at this stage to rank and/or screen alternatives, ruling out the less promising and keeping the best. Besides, the complexity of both, MOO and MCDM, increases with the number of objectives. In practice, the visualization and analysis of the Pareto set becomes highly difficult in problems with more than three objectives. Dimensionality reduction techniques aim at overcoming these limitations by identifying redundant objectives that can be omitted while still preserving the problem structure to the extent possible (see sections 3.2.2 and 7.3). As will be explained in detail later in this thesis, in this work we developed an efficient objective reduction algorithm to expedite the solution of MOO problems arising in PSE.

2.2.1 Objective reduction

Objective reduction identifies redundant objectives that can be eliminated from the MOO model, thereby expediting its solution. Different approaches have been proposed so far for objective reduction in MOO. Deb and Saxena (2005) proposed a method based on principal component analysis for reducing the number of objectives in MOO. More recently, Thoai (2011) proposed an approach to reduce the number of criteria as well as the dimension of a linear MOO problem using the concept of representative and extreme criteria, while López Jaimes et al. (2009) introduced two algorithms

for objective reduction in MOO based on a feature selection technique.

In a seminar work, Brockhoff and Zitzler (2006c) analyze in mathematical terms the issue of objective reduction in MOO. They introduced the concept of delta error in MOO, defined as the approximation error that arises after removing objectives in a MOO problem. These authors formally stated the following two problems:

1. computing a minimum objective subset (MOSS) of a multi-objective problem that does not exceed a certain approximation error (denoted as the δ-MOSS problem); and

2. identifying a minimum objective subset of size k with minimum approximation error (k-MOSS problem) (see section 7.4).

Based on these ideas, Guillén-Gosálbez (2011a) introduced an approach for dimensionality reduction based on an MILP that solves both the k-MOSS and δ-MOSS problems, and which takes advantage of the powerful branch-and-cut algorithms available for MILPs. In this thesis, we present a new decomposition approach for objective reduction based on the work by Guillén-Gosálbez (2011a) that is shown from numerical examples to outperform other existing algorithms for objective reduction (Brockhoff and Zitzler, 2006c).

2.2.2 The ϵ-constraint method

The ϵ-constraint method is an algorithm widely used to solve multi-objective optimization (MOO) problems. It was first introduced by Haimes, Y.Y.;

Lasdon, L.S.; Wismer (1971), and it is based on solving a set of single-objective problems in each of which one objective is kept as main objective while the rest are transferred to auxiliary constraints that impose bounds on them. This algorithm has found many applications in PSE, including the optimal design of supply chains (Yue et al., 2014), refineries (Gebreslassie et al., 2013; Zhang et al., 2014; Santibañez Aguilar et al., 2014; Murillo-alvarado and El-halwagi, 2013), reverse osmosis networks (Du et al., 2014), and the optimization of metabolic pathways in system biology (Pozo and Guillén, 2012).

The main limitation of this method is that its computational burden grows rapidly in size with the number of objectives. Furthermore, the traditional ϵ-constraint algorithm requires the definition of a set of epsilon parameters values, which are typically generated by splitting the domain of each objectives in equidistant intervals. This leads to falling repeatedly into the same solution or an unfeasible one, thus wasting large computational efforts. To avoid this, as will be described in detail later in this thesis, we propose here several improvements for the ϵ-constraint algorithm based on its integration with rigorous dimensionality reduction methods and pseudo/quasi-random sequences.

Part II

Second Part: Research Work

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

Chapter 3

Methodology

We next review the main general mathematical programming methods before presenting in detail the algorithms developed in the context of this thesis (which are based on the former).

3.1 Overview optimization

Several solution methods have been proposed so far for dealing with the complexity associated with PSE problems. We next provide an overview of these strategies, with emphasis on optimization tools based on mathematical programming and metaheuristics.

3.1.1 Mathematical programming

Mathematical programming aims to find an optimal solution of a given a optimization problem subject to a set of constraints. Optimization problems can be classified according to the type of variables they contain (continuous and, discrete variables). Problems that only include continuous variables belong to linear programming (LP) and non-linear programming (NLP) (depending on whether the constraints are all linear or contain at least

one nonlinear equation). Problems with discrete variables are modelled as either mixed-integer linear programming (MILP) and mixed-integer nonlinear programming (MINLP) models. Dynamic optimization (problems including differential equations), and stochastic programming (optimization under uncertainty) and other types of mathematical models encountered in mathematical programming.

The aforementioned problems can be solved by different techniques. In the past, most of them were based on a trial and error method. In the recent past, a wide variety of systematic approaches have emerged to perform this task.

In this introductory section, we start by considering a general constrained optimization problem (Bazaraa et al., 2006):

$$\begin{aligned} min \quad & f(x,y) \\ s.t \quad & h(x,y) = 0 \\ & g(x,y) \leq 0 \\ & x \in X \\ & y \in Y \end{aligned} \tag{3.1}$$

Hereby, $f(x,y)$ is the objective function, x is the set continuous variables, and y are the integer variables, $h(x,y) = 0$ are the equality constraints and $g(x,y) \leq 0$ are the inequality constraints. Any optimization problem can be represented in this form. Note that maximizing function $f(x,y)$ is equivalent to minimizing $-f(x,y)$. Moreover, if we have inequal-

ities greater than zero, these may be transformed by multiplying the two terms by minus one.

3.1.1.1 Linear programming (LP)

Linear programming (LP) problems contain only linear functions and continuous decision variables (Dave et al., 1998):

$$\begin{aligned} min \quad & Z = c^T \cdot x \\ s.t \quad & A \cdot x = b \\ & C\dot{x} \leq d \\ & x \geq 0 \end{aligned} \tag{3.2}$$

The standard solution method to solve LP problems is the simplex method (Dantzig, 1963), although in the last decades interior point methods (Wu et al., 2002) have been extensively used for highly constrained LP problems (e.g., problems with around 100.000 constraints and variables). The major commercial solvers developed for LP optimization problems are CPLEX and OSL, which are based on the simplex method, and XPRESS, which makes use of a Newton barrier interior point method.

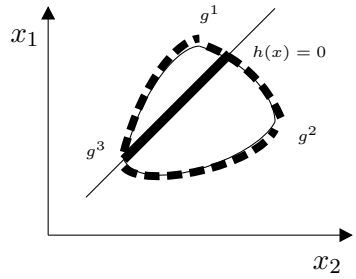


Figure 3.1: Feasible region for three inequalities and one equation.

3.1.1.2 Non-linear programming (NLP)

Non-linear programming (NLP) problems contain at least one non-linear equation and only continuous variables (not a single discrete variable).

$$\begin{aligned} min \quad & f(x) \\ s.t \quad & h(x) = 0 \\ & g(x) \leq 0 \\ & x \in X \end{aligned} \tag{3.3}$$

Two major methods for NLP optimization are used: successive quadratic programming (SQP) (Powell, 1978) and the reduced gradient method (Murtagh and Saunders, 1978). In the SQP algorithm, the basic idea is to solve a quadratic programming subproblem at each iteration. In contrast, the reduced gradient method solves a sequence of subproblems with linearized constraints. SQP generally requires less iterations than the gradient method. However, for large-scale problems, the reduced gradient method tends to be more robust.

The main SQP solvers are SNOPT, KNITOR, IPOPT and rSQP, whereas GRG2, CONOPT and MINOS are the main solvers based on the reduced gradient method .

3.1.1.3 Mixed integer-linear programming (MILP)

This type of problem model is an extension of LP where some of the variables may adopt an integer value. The general form of an MILP problem is the following:

$$\begin{aligned} min \quad & Z = c^T \cdot x + b^T \cdot y \\ s.t \quad & A \cdot x + B \cdot y \leq d \\ & x \geq 0 \\ & y \in \{0,1\}^m \end{aligned} \tag{3.4}$$

The principal method for solving MILPs is the LP-based branch and bound (Nemhauser and Wolsey, 1988). This technique is baed on a tree enumeration where at each node a relaxed LP subproblem is solved. Another technique for MILP optimization makes use of cutting planes, a method based on generating cuts from the LP relaxation. Currently, most of the commercial solvers combine both techniques. The most widely used computer packages implementing these combined methods are CPLEX, OSL, LINDO and ZOOM.

3.1.1.4 Mixed integer-nonlinear programming (MINLP)

This type of problem model presents features of both NLPs and MILPs. The variables may adopt either real or integer values and some of the con-

strains or the objective function may be non-linear. The general form of a MINLP problem is the following:

$$\begin{aligned} min \quad & Z = c^T \cdot x + b^T \cdot y \\ s.t \quad & A \cdot x + B \cdot y \leq d \\ & x \in X, y \in Y \\ X = \{x \mid x \in R^n, x^L \leq x \leq x^U, B \cdot x \leq b\} \\ Y = \{y \mid y \in \{0,1\}^m, A \cdot y \leq a\} \end{aligned} \tag{3.5}$$

There are several solution algorithms: branch and bound (Gupta and Ravindran, 1985), branch and cut (Stubbs and Mehrotra, 1999), generalized benders decomposition (Geoffrion, 1972) and outer-approximation (Duran and Grossmann, 1986).

3.1.1.5 Mixed linear fractional programming (MILFP)

A mixed-integer linear fractional program (MILFP) is a special type of non-convex MINLP, which includes both continuous and discrete variables and which seeks to optimize an objective function that is expressed in general form as the ratio of two linear functions subject to linear constraints. A general MILFP can be formulated as:

$$\begin{aligned} \max \quad & \frac{A_0 + \sum_{i \in I} A1_i x_i + \sum_{j \in J} A2_j y_i}{B_0 + \sum_{i \in I} B1_i x_i + \sum_{j \in J} B2_j y_j} \\ s.t \quad & C_{0k} + \sum_{i \in I} C1_{ik} x_i + \sum_{j \in J} C2_{jk} y_j = 0, \forall k \in K, \\ & x_i \geq 0, \forall i \in I \\ & y_i \in \{0,1\}, \forall j \in J \end{aligned} \tag{3.6}$$

where x_i are continuous variables and y_j are discrete variables. In this problem 3.6, it is assumed that the denominator $B_0+\sum_{i\in I} B1_i x_i+\sum_{j\in J} B2_j y_j > 0$. Thus, problem 3.6 includes $|I|$ continuous variables, $|J|$ binary variables, and $|K|$ equality constraints (all of which are linear). The readers can find more details in Yue et al. (2013).

3.1.1.6 Multi-objective optimization (MOO)

Multi-objective optimization (MOO) problems include more than one objective function, and can generally be posed as follows:

$$\begin{aligned} &\min_x F(x) := (f_1(x), ..., f_k(x)) \\ &\text{s.t. } g_j(x) \leq 0, && j = 1, 2, \ldots, m, \\ &\quad h_l(x) = 0, && l = 1, 2, \ldots, e, \end{aligned} \tag{3.7}$$

where k is the number of objective functions, m is the number of inequalities constraints, and e is the number equalities constraints. $x \in E^n$ is the vector of *design space* or *decision variables*, while n represents the number of independent variables x_i. $F(x) \in E^k$ is a vector of objective functions $F_i(x) : E^n \longrightarrow E$, where $F_i(x)$ is the value function. The feasible design space X is defined as the set $\{x \mid g_j(x) \leq 0, j = 1, 2, \ldots, m;\ h_i(x) = 0, i = 1, 2, \ldots, e\}$. The feasible criterion space Z is defined as the set $\{F(x)|x \in X\}$. There are several methods to deal with MOO (Marler and Arora, 2004). We present in this thesis the approach developed in section 3.2.3.

The solution to MOO problems is not a single point but a set of so-

lutions which form the so-called Pareto front. These solutions hold the property of being equally optimal, as it is not possible to improve one of their objectives without worsening at least one of the others. This property is formulated as follows:

A solution x^* is called Pareto optimal if there is no other x that dominates x^* with respect to the set of all of the objectives, where x (weakly) dominates y ($x \preceq_F y$) with respect to the objective set F, if $x_i \leq y_i$ in each objective $i \in F$.

The quality of the Pareto front is assessed by the hypervolume indicator (or S-metric, Lebesgue measure), which was introduced by Zitzler and Thiele (1998). This metric, is regarded as a fair measure of the hyperspace that is dominated by at least one point of the Pareto set. The dominated hypervolume is calculated with respect to a reference point which is chosen to coincide with the nadir point.

3.1.2 Incomplete algorithms: heuristics and metaheuristics

In this section we introduce heuristic approaches for solving optimization problems, then we delve into algorithms that are known as metaheuristics. In fact, the hybridization of metaheuristics and integer linear programming is an important aspect of this work. Therefore, at the end of this section we will give a brief description of the hybridization of metaheuristics with other optimization techniques (Glober and Kochenberger, 2003).

3.1.2.1 Combinatorial optimization

A combinatorial optimization problem (COP), following the definition of Papadimitriou and Steiglitz (C.H. Papadimitriou and K. Steiglitz, 1982), is the pair $P = (S, f)$ consisting by the finite set of objects S and an objective function $f : S \to \mathbb{R}^+$, which assigns a positive value to each of the objects of S. The goal is to find an $s \in S$ which has a cost value $f(s)$ that is lower than (or equal to) the cost value of any other object in S. A well-known example of a COP is the Traveling Salesman Problem (TSP) (Lawler et al., 1985), which is defined as follows.

Definition In the Traveling Salesman Problem (TSP) (Shmoys et al., 1985) a complete graph $G = (V, E)$ with a weight $w_e \in \mathbb{R}^+$ for each edge $e \in E$ is given. The goal is to in find the minimum Hamiltonian cycle in G. The objective function value $f(s)$ is calculated as the sum of the weights of the edges that form the Hamiltonian cycle s, and hence the search space S consists of all possible Hamiltonian cycles that exist in G.

Note that each COP can be modelled as an MILP in different ways. The resolution of any COP, (or any computer problem) is performed through an algorithm. Algorithms may be *complete*, *probabilistic* or *approximate*: complete methods guarantee to find an optimal solution. However, when the size of the problem instance increases, the computation time required by complete methods may be impractically high. In the case of COP problems that are NP-hard, no polynomial algorithm exists to solve them. Therefore, when rather large instances of NP-hard problems are concerned, approximate algorithms are often the only alternative. While these methods produce good solutions in a reasonable amount of computation time, they

do not guarantee to find optimal solutions. There are two basic types of approximate algorithms, constructive heuristics and local search.

3.1.2.2 Constructive heuristics

Constructive heuristics are the most typical approximate algorithms for solving COP. These algorithms build solutions from scratch, starting from an empty initial solution. They employ a *construction mechanism* which will add solution components at each step, according to a cost function, until the solution is completed or the process is stopped by another criterion.

The schema of a constructive heuristic is shown in algorithm 1. The algorithm is initialized with an empty partial solution s^p. Given a partial solution s^p, a set of solution components $cc(s^p)$ can be derived for the extension of s^p. A greedy value $\eta(c)$ is assigned to each component $c \in cc(s^p)$ which serves as a selection criterion: at each step we choose the component $c \in cc(s^p)$ with the maximal greedy value and extend the partial solution s^p by adding c. This process is repeated until set $cc(s^p)$ is empty, or until some other criterion indicates that the solutions construction process should be stopped.

3.1.2.3 Local search methods

These algorithms start from some initial solution and iteratively try to replace the current solution by a better one, looking in a neighborhood formally defined as follows:

Definition A neighborhood is a function $N : S \rightarrow 2^S$ that assigns to each

Algorithm 1 Constructive Heuristic

$s^p = \emptyset$
Generate($cc(s^p)$)
while $cc(s^p) \neq \emptyset$ **do**
  c = Selection($Cc(s^p)$)
  s^p = extend s^p by adding the component c
  Generate($cc(s^p)$)
end while

$s \in S$ a set of neighboring solutions $N(s) \subseteq S$. Hereby, each solution $s' \in N(s)$ is obtained by applying an operator to s, which applies a rather small change to s and hereby creates a new solution s'. Operators or movements are applied in a particular order.

A neighborhood with an instance of the problem defines a search space, which can be represented by a graph where the vertices are solutions that are labeled by the value of the objective function, and the arcs represent the neighborhood relationship of the solutions. A solution $s^* \in S$ is called *globally minimal solution* if for all $s \in S$ it holds that $S : (f(s^*) \leq f(s))$. The introduction of a neighborhood structure enables us to additionally define the concept of *locally minimal solutions*.

Definition A local minimum with respect to a neighborhood N is a solution $\widehat{s}$ such that $\forall s \in N(\widehat{s}) : f(\widehat{s}) \leq f(s)$. And $\widehat{s}$ is a strict local minimum if $\forall s : s \in N(\widehat{s}) : (f(\widehat{s}) < f(s))$.

The simplest method of local search is the *iterative improvement local search* where at each step a neighbor is chosen which is better than the current solution. This method is outlined in algorithm 2.

Algorithm 2 Iterative Improvement Local Search

$s = generate\_initial\_solution()$
while $\exists s' \in N(s)$ such as $f(s') < f(s)$ **do**
  $s = choose\_improving\_neighbor(N(s))$
end while

There are mainly two ways to implement function *choose_improving neighbor*$(N(s))$: searching the neighborhood in a pre-defined order returning the first neighbor that is better than the current solution, or performing an exhaustive search through the neighborhood and returning the best neighbor.

3.1.2.4 Metaheuristics

Metaheuristic (Glover, 1986; Reeves, 1993) were introduced in the 70's. They are approximate algorithms that combine basic heuristics to explore the search space more effectively and efficiently than constructive heuristics and local search. The term metaheuristic is a Greek compound word. *Heuristic* comes from *heuriskein* which means "search", while the suffix *meta* means " beyond" referring to a higher level. There are different definitions of metaheuristics according to the respective authors. Here is a representative one:

"A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a

construction method." (Voss et al., 1999)

Here we give an excerpt of some of the fundamental properties taken from different authors, such as for example Blum (2005). Metaheuristics are characterized by:

- Metaheuristics are strategies that "guide" the search process.
- The goal is to efficiently explore the search space in order to find (near)optimal solutions.
- Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes.
- Metaheuristic algorithms are approximate and usually non-deterministic.
- They may incorporate mechanisms to avoid getting trapped in confined areas of the search space.
- The basic concepts of metaheuristics can be described on an abstract level (i.e., not tied to a specific problem)
- Metaheuristics are not problem-specific.
- Metaheuristics may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper level strategy.
- Advanced metaheuristics use search experience (embodied in some form of memory) to guide the search.

3.1.2.5 Classification of Metaheuristics

Metaheuristics may be classified in different ways. In the following paragraphs we outline some of these possible classifications (Blum, 2004; Glober and Kochenberger, 2003):

Nature-inspired vs. non-nature inspired. Probably the most intuitive way of classifying metaheuristics refers to their origins. There are nature-inspired algorithms, such as evolutionary algorithms and ant colony optimization, and non nature-inspired ones such as tabu search and iterated local search. However, this classification may not be very meaningful. Many recent hybrid algorithms can not be assigned to any of the two classes. Moreover, it is sometimes difficult to clearly attribute an algorithm to one of the two classes.

Single point vs. population-based search. Another characteristic that can be used for the classification of metaheuristics is the number of solutions that a metaheuristic works on at the same time: does the algorithm make use of a population or only of a single solution at any time. Algorithms that work on single solutions are generally referred to as *trajectory methods*. They comprise all metaheuristics that are based on local search. This is because their search process describes a trajectory in the search space. Population-based metaheuristics, on the contrary, either perform search processes which can be described as the evolution of a set of points in the search space, or they perform search processes which can be described as the evolution of a probability distribution over the search space.

Dynamic vs. static objective function. Metaheuristics can also be classified regarding the way in which they use the objective function. While some algorithms use static objective functions during run-time, some others, such as guided local search, modify the objective function during the search. The idea behind this approach is to escape from local minima by modifying the search landscape.

One vs. various neighborhood structures. Many metaheuristic algorithms only use one single neighborhood structure. In other words, the search landscape topology does not change in the course of the algorithm. Other metaheuristics, such as variable neighborhood search, use more than one neighborhood structure. This opens the possibility to diversify the search by swapping between different search landscapes.

Memory-based vs. memory-less methods. A very important feature to classify metaheuristics is based on whether they use memory or not. Memory-less algorithms perform a Markov process, as the information they exclusively use to determine the next action is the current state of the search process. Usually we differentiate between the use of short term and long term memory. The former usually refers to recently performed moves, visited solutions or, in general, decisions taken. The latter is usually an accumulation of synthetic parameters about the search. The use of memory is nowadays recognized as one of the fundamental elements of a powerful metaheuristic.

3.1.2.6 Trajectory Methods

As mentioned before, trajectory methods are characterized by the fact that the work on a single solutions at any time. Their search process describes a trajectory in the search space.

3.1.2.6.1 Simulated Annealing The idea of Simulated Annealing (SA) (Metropolis et al., 1953) is taken from the metallurgical industry. It is based on the process of cooling down metal and glass, slow enough in order to get (nearly) perfect crystal structures. In each iteration, the SA selects a solution $s' \in N(s)$ at random, where s is the current solution. If $f(s') < f(s)$ then we replace s by s'. Otherwise, s is still replaced by s' with probability $p(s' \mid T_k, s)$ (equation 3.8) which follows a Boltzmann distribution:

$$p(s' \mid T_k, s) = e^{-\frac{f(s')-f(s)}{T_k}} \quad , \tag{3.8}$$

where T_k is a so-called temperature parameter. This version of SA works without memory, but the use of memory for storing the history can be beneficial. The framework of the method is outlined in Algorithm 3.

Now the functions that are used in the SA algorithm are explained in more detail:

- *generate_initial_solution*()**:** the algorithm starts by generating a solution that can be random solution or the result of a constructive heuristic.

Algorithm 3 Simulated annealing (SA)

$s = generate\_initial\_solution()$
$k = 0$
$T_k = set\_initial\_temperature()$
while end condition not found **do**
  $s' = choose\_neighbor\_at\_random(N(s))$
  if $f(s') < f(s)$ **then**
    $s = s'$
  else
    Accept s' as new solution with probability $p(s' \mid T_k, s)$ (see 3.8)
  end if
  $update\_temperature(T_k, s)$
end while

- $set\_initial\_temperature()$**:** the initial temperature has to be one that allows to move to considerably worse solutions at the beginning of the algorithm. Its setting is crucial for the success of the algorithm.

- $choose\_neighbor\_at\_random(N(s))$**:** randomly select a neighbor.

- $update\_temperature(T_k)$**:** the temperature parameter needs to be updated at each iteration. The idea is to reduce the value of this parameter step by step in order to gradually decrease the probability to move to worse solutions. An example is the following scheme:

$$T_k = T_{k-1} \cdot \alpha, \ \alpha \in (0, 1) \tag{3.9}$$

  The temperature planning method is crucial for obtaining good re-

sults.

3.1.2.6.2 Tabu Search The simple version of tabu search algorithm (TS) (Glover, 1986) (see Algorithm 4) is based on the (*best-improvement*) version of local search and uses short term memory to avoid local minima and cycling. The short-term memory is implemented by a *tabu list*, denoted as TL, which keeps track of the latest solutions visited. $N_a(s)$ is called the *allowed set*. This set is a subset of the neighborhood $N(s)$, generated by eliminating the solutions stored in TL. At each iteration we choose the best solution of all the ones in $N_a(s)$ as new current solution. This solution is also stored in TL by function $update(TL, s, s')$. If TL exceeds its maximum capacity, then the oldest solution is removed from TL. This means that TL is managed following a first-in first-out (FIFO) policy. As we can see the algorithm terminates when the end condition is true or when $N_a(s)$ is empty.

Algorithm 4 Simple tabu search (TS)

$s = generate\_initial\_solution()$
$TL = \emptyset$
while end condition not true **do**
  $N_a(s) = N(s) \setminus TL$
  $s' = argmin\{f(s'') \mid s'' \in N_a\}$
  $update(TL, s, s')$
  $s = s'$
end while

The implementation of a short-term memory using a list that stores complete solutions is not practical because keeping a list of solutions is

very inefficient. Therefore, rather than storing solutions, we keep the *solution components* involved in the movements. So we need a TL for each type of solution component. Each tabu list TL will be involved in defining a *tabu condition* that serves to filter the neighborhood of the current solution. However, note that storing solution components instead of complete solutions, even though it is much more efficient, has a potential loss of information. This is because when forbidding visiting all solutions that contain this component, solutions that were not visited before might may be forbidden. To solve this problem, *aspiration criteria* are used. They may allow to include otherwise forbidden solutions into set N_a. The most common aspiration criterion used is when a solution is better than the best solution found. Such a solution should, of course, not be forbidden. Algorithm 5 shows the framework of this more practical version of TS.

Algorithm 5 Tabu search (TS)

$s = generate\_initial\_solution()$
$init\_tabu\_lists(TL_1, \ldots, TL_r)$
while end condition not found **do**
  $N_a(s) = \{s' \in N(s) \mid s'$ is not forbidden,
or aspiration conditions are satisfied$\}$
  $s' = argmin\{f(s'') \mid s'' \in N_a\}$
  $update\_tabu\_lists(TL_1, \ldots, TL_r, s, s')$
  $s = s'$
end while

3.1.2.6.3 Explorative Local Search Methods In this section we present some other trajectory methods, which are *greedy randomized adaptive search procedures (GRASP)*, *variable neighborhood search (VNS)*, *guided*

local search (GLS) , and *iterate local search (ILS).*

1. **Greedy Randomized Adaptive Search Procedures** The greedy randomized adaptive search procedure Feo and Resende (1995) is a metaheuristic that combines a constructive metaheuristic with local search (see Algorithm 6). GRASP is an iterative procedure that consists of two phases: the construction of a solution (see Algorithm 7) and the improvement of the solution built.

Algorithm 6 Greedy randomized adaptive search procedure (GRASP)

while end condition not found **do**
  $s = build\_greedy\_random\_solution()$
  $apply\_local\_search(s)$
end while

The construction method randomly creates a solution s^p step by step, by adding components of a finite list called restricted candidate list (RCL). The RCL is made up of the first α components from $cc(s^p)$, assuming that the elements of $cc(s^p)$ are ordered by a Greedy function η. Note that α is an important parameter.

Algorithm 7 Greedy randomized solution construction

$s^p = \langle\rangle$
$\alpha = determine\_length\_of\_candidate\_list()$
while end condition not found **do**
  $RCL = generate\_candidate\_list(\eta, cc(s^p), \alpha)$
  $c = choose\_at\_random(RCL)$
  $s^p =$ extend s^p by adding solution component c
end while

As local search it is possible to use any of the available local search

algorithms, such as the simple search algorithms, or more advanced methods such as SA or TS. To be effective, GRASP has to satisfy (at least) two conditions:

- The constructive heuristic should explore the best areas of the search space.
- Built solutions should belong to different local minima of the utilized local search.

In order to satisfy these two conditions the algorithm components have to be properly chosen. Moreover, the length of the candidate list must be adequately chosen.

2. **Variable Neighborhood Search**

Variable neighborhood search (Hansen and Mladenovi, 2001) applies strategies to switch between different neighborhoods of a finite set of predefined neighborhoods (see Algorithm 8):

VNS is initialized with a set of neighborhoods that are required to meet the following condition: $\forall s : s \in S : (|N_1(s)| < |N_2(s)| < \ldots < |N_{kmax}(s)|)$. Second, an initial solution s is generated. Then the outer loop of the algorithm iterates until the stopping conditions are reached. Within the outer loop, the neighborhood index k is initialized to 1. Each iteration of the inner loop has three phases: *shaking*, *local search* and *acceptance* of a new current solution. In the shaking phase a solution s' of the $k-$th neighborhood $N_k(s)$ of s is chosen, and is then subject to the local search phase which results in

a local minimum s''. If $f(s'') < f(s)$ then we replace s by s'', and initialize the neighborhood index k to 1. However, if $f(s'') \geq f(s)$ then we increase the neighborhood index k by one in order to diversify the search process.

Algorithm 8 Variable neighborhood search (VNS)

```
define neighborhoods N_k, k = 1, ..., k_max
s = generate_initial_solution()
while end condition not reached do
    k = 1
    while k < k_max do
        s' = choose_at_random(N_k(s))
        s'' = local_search(s')
        if f(s'') < f(s) then
            s = s''
            k = 1
        else
            k = k + 1
        end if
    end while
end while
```

3. Guided local search

Guided local search (GLS) (Voudouris and Tsang, 1999) is a metaheuristic that uses a dynamic objective function to escape from local minima. The dynamic objective function f' is obtained from an adaptive change of the original objective function f. This change is based on a set of, in general, m characteristics of a solution: $sf_i, i = 1, \ldots, m$. These features can be used to differentiate solutions. $I(i, s)$

tells us if the property sf_i is present in the solution s.

$$I(i,s) = \begin{cases} 1 : \text{if the feature } sf_i \text{ is in the solution } s \\ 0 : \text{otherwise} \end{cases} \tag{3.10}$$

During the execution of the algorithm, the original function $f(\cdot)$ is replaced by $f'(\cdot)$, which is obtained from $f(\cdot)$ by adding the penalty $p_i : i = 1, \ldots, m$, where $\lambda > 0$ is the influence of p_i in $f'(\cdot)$:

$$f'(s) = f(s) + \lambda \sum_{i=1}^{m} p_i \cdot I(i,s) \tag{3.11}$$

The algorithm (see Algorithm 9) works as follows. First, an initial solution s is generated and the vector of penalties $\mathbf{p}$ is initialized to all zeros. Then, at each iteration local search is applied to the current solution s based on the changed objective function f'. This results in a solution s'. Depending on s' the penalty vector $p = (p_1, \ldots, p_m)$ is modified in the function $update\_vector\_penalty(p, s')$ calculating the *utility* $U(i, s')$ for each property:

$$U(i,s') = I(i,s') \cdot \frac{c_i}{1 + p_i} \tag{3.12}$$

The elements p_i of the penalty vector $\mathbf{p}$ may be modified as follows:

$$p_i = p_i + 1 \tag{3.13}$$

However, there are also other ways for the penalty vector update.

Algorithm 9 Guided local search (GLS)

$s = generate\_initial\_solution()$
$\mathbf{p} = (0, \ldots, 0)$
while end condition not found **do**
  $s' = local\_search(s, f')$
  $update\_penalty\_vector(\mathbf{p}, s')$
  $s = s'$
end while

4. **Iterated local search**

In each iteration of iterated local search (ILS) (Stützle, 1999) (see Algorithm 10) the current solution s', which is a local minimum, is subject to the perturbation function, which returns a perturbed solution s''. Afterwards, solution s'' is subject to local search which provides a new local minimum s'''. Finally, the algorithm must decide between solutions s' and s''' for a new current solution. The *perturbation* prevents the algorithm from being trapped in local minima, whereas the *acceptance criterion* has an influence on the diversification and intensification behavior of the algorithm.

Algorithm 10 Iterated local search (ILS)

$s = generate\_initial\_solution()$
$s' = local\_search(s)$
while end condition not found **do**
  $s'' = perturbation(s', memory)$
  $s''' = local\_search(s'')$
  $s' = apply\_acceptance\_criteria(s''', s', memory)$
end while

- The term *memory* (see Algorithm 10) refers, for example, to the fact that solutions found during the search process may be stored and used for different purposes.

- *generate_initial_solution*(): this function constructs the initial solution. The most important requirement is to be fast, such as, for example, the generation of a random solution or the use of a simple greedy heuristic.

- *perturbation*($s', memory$): the perturbation usually tends to be non-deterministic. The most important feature of the perturbation is the *strength*, defined as the "damage" inflicted on the current solution. This feature can be fixed or variable. In the first case the distance between s' and s'' will always be the same regardless of the state of the search process. However, a *variable* strength is usually more effective and must be experimentally found, depending on the current state of the search process.

- *apply_acceptance_criteria*($s''', s', memory$): the two extreme cases for the acceptance criterion are as follows. We may only accept s''' as new current solution in case it is better than s', or we may always accept the new local minimum regardless of its quality. In between these two cases we have several possibilities to adopt, for example, acceptance criteria similar to the one used in SA.

3.1.2.7 Population-Based Methods

The most well known metaheuristics based on populations for the application to combinatorial optimization problems are evolutionary algorithms (EAs), or (*evolutionary computation*(EC) algorithms), and *ant colony optimization (ACO)* algorithms.

3.1.2.7.1 Evolutionary Algorithms Evolutionary algorithms (EC) are inspired by nature's ability to adapt to the environment, that is, the natural evolution of species. In each iteration the individuals that make up the current population are subject to operations such as recombination, which produces the individuals of the next generation (iteration). These next generation individuals are selected based on their fitness, which is determined by their objective function value.

The family of evolutionary algorithms can be divided into three categories of independent development: Firstly, evolutionary programming (EP) was introduced by Fogel (Fogel et al., 1966). Then, evolutionary strategies (ES) were proposed by Rechenberg (Rechenberg and Eigen, 1973) and finally genetic algorithms (GA) as proposed by Holland (Holland, 1992).

Algorithm 11 shows the basic structure of EC algorithms. In the algorithm, P denotes the population. New individuals are produced by applying recombination and mutation operators to the individuals of P. Then the new population P'' is selected from P and from the newly generated individuals. The main features of the EC algorithm are as follows:

Algorithm 11 Evolutionary Computation (EC)

$P = generate\_initial\_population()$
$evaluate(P)$
while end condition not found **do**
  $P' = recombination(P)$
  $P'' = mutation(P')$
  $evaluate(P'')$
  $P = choose(P'', P)$
end while

- **Representation of the individuals**: commonly used solution representations are bit-strings or permutations of integers. Note that a solution representation is generally called *genotype* whereas the solution which is represented is called *phenotype*.

- **Process of evolution**: in each iteration, individuals are chosen to constitute the next generation. In some cases the new population is exclusively composed of new individuals. In other cases, the new population is chosen as the best set of solutions from the old population and the newly generated individuals. Many times the number of individuals is constant from iteration to iteration.

- **Neighborhood function**: in the context of EC algorithms, this concept refers to a function $N_{\varepsilon c} : I \longrightarrow 2^R$ that assigns to each individual $i \in I$ a set of individuals $N_{\varepsilon c}(i) \subseteq I$, which refers to the set of solutions with which individual i can be recombined.

- **Information source**: this refers to the input parameters of a recombination operator. In the standard case, a pair of parent individuals

are recombined to generate one or two new individuals (children). But it could also be the case that more than two individuals are recombined to create new individuals.

- **Unfeasible solutions**: an important aspect of EC algorithms is the way of dealing with individuals who are not feasible. This problem is often encountered, because many genetic operators may generate unfeasible individuals. There are three ways to address this: the first is to discard the infeasible individuals, the second way is to penalize infeasible individuals based on decreasing their quality, and the third way is to try to fix the corresponding individual.

3.1.2.7.2 Ant Colony Optimization In order to solve a given CO problem, the metaheuristic ant colony optimization (ACO) has first to derive a finite set C of solution components, which is used to assemble solutions (Dorigo and Blum, 2005; Blum and Roli, 2003). Then we have to define a set T of pheromone values, which is generally known as the *pheromone model*. Together with a mechanism for constructing solutions, these values define a probability distribution over the search space. The pheromone model is the heart of each ACO metaheuristic. Generally, each solution component from C has an associated pheromone value $\tau_i \in T$, so that solutions can be generated by assembling components probabilistically.

The ACO in each iteration has two phases:

- Solutions are built on the basis of the pheromone model.

- The values of the pheromone model are updated depending on the quality of the solutions built.

The general idea is that the pheromone model can guide the search process to those parts of the search space containing high quality solutions. This is because the pheromone value update increases the value of the pheromone components depending on the quality of the corresponding solutions. A precondition for ACO is that good solutions contain good solution components.

Algorithm 12 Ant Colony Optimization (ACO)

```
while end condition not found do
    build_solution()
    update_pheromones()
    daemon_actions()
end while
```

In the following we give a more detailed description of ACO (see Algorithm 12). ACO is an iterative algorithm which consists of three stages or procedures *build_solution*(), *update_pheromones*(), and *daemon_actions*(). These procedures are explained in more detail next:

build_solution(): (see Algorithm 13) the exploration of the ants is simulated with a probabilistic constructive heuristic that assembles solution components from a finite set $C = \{c_1, \ldots, c_n\}$. Each component c_i of this set C has an associated pheromone value τ_i. A complete solution is obtained as a sequence of solution components s. A solution construction start with the empty sequence $s = \langle\rangle$. At each step, the current sequence s is ex-

tended by adding a solution component from $N(s) \subseteq C \setminus s$. The selection of component $c_i \in N(s)$ is done in a probabilistic way: to each component we assign a probability $p(c_i \mid s)$ which depends on greedy information (η) and pheromone information:

$$p(c_i \mid s) = \frac{[\tau_i]^\alpha \cdot [\eta(c_i)]^\beta}{\sum_{c_j \in N(s)} [\eta_j]^\alpha \cdot [\eta(c_j)]^\beta}, \forall c_i \in N(s), \tag{3.14}$$

Hereby, η is a greedy function which is also known as the *heuristic information*. This greedy function assigns to each component $c_i \subseteq N(s)$ a value $\eta(s)$. Moreover, the positive exponents α and β are parameters that scale the weight of the heuristic information in relation to the weight of the pheromone information.

Algorithm 13 *build_solution*()

```
s = ⟨⟩
compute(cc(s))
while Cc(s) ≠ ∅ do
    c = choose(cc(s))
    s = add component c to s
    compute(cc(s))
end while
```

update_pheromones(): the pheromone updating process has two phases: the first phase consists of the so-called pheromone evaporation, which decreases the value of the pheromones uniformly. This function is necessary in order to avoid the rapid convergence of the algorithm and explore different regions of the solution space. Second, pheromones are increased in

each iteration as follows:

$$\tau_i = (1-\rho)\cdot\tau_i + \rho\cdot \sum_{\{s\in S_{upd}|c_i\in s\}} (w_s\cdot F(s)) \ , \tag{3.15}$$

for $i = 1, \ldots, n$. Hereby, S_{upd} the set of solutions that are used to update the pheromones, and $F(s)$ is a function $F : S \longmapsto \mathbb{R}^+$ such that $f(s) < f(s') \longrightarrow F(s) \geqslant F(s'), \forall s, s' \in S$, and $w_s \in \mathbb{R}^+$ denotes the weight of the solution s. In most cases, S_{upd} is composed of the best solutions built in the current iteration.

daemon_action(): here we apply those actions that require a global vision.

3.1.2.8 Hybrid Metaheuristics

The concept of hybrid metaheuristics was developed in recent years, even if the idea of combining different metaheuristic strategies and algorithms dates back to the 1980s. Today, we can observe a generalized common agreement on the advantage of combining components from different search techniques and the tendency of designing hybrid techniques is widespread in the fields of operations research and artificial intelligence. The consolidated interest around hybrid metaheuristics is also demonstrated by publications on classifications, taxonomies and overviews on the topic (Raidl, 2006; Talbi, 2002).

In general, a hybrid metaheuristic is obtained by combining a metaheuristic with algorithmic components originating from other techniques

for optimization (possibly another metaheuristic). We may distinguish between two categories: the first consists in designing a solver including components from a metaheuristic into another one, while the second combines metaheuristics with other techniques typical of fields like operations research and artificial intelligence. A prominent representative of the first category is the introduction of trajectory methods into population based techniques or the use of specific local search methods into a more general trajectory method such as ILS. The second category includes hybrids resulting from the combination of metaheuristics with constraint programming (CP), integer programming (IP), tree-based search methods, data mining techniques, etc. Both categories contain numerous instances and an exhaustive description is out the scope of this Thesis Blum et al. (2011).

3.1.2.8.1 Large Neighbourhood Search The large neighbourhood search (LNS) was introduced by (Shaw, 1998) and it is an example of hybrid metaheuristic that we apply in this thesis to solve supply chain problems (see Section 6). The main idea behind the LNS hybrid metaheuristic is that the large neighborhood allows the algorithm to explore the solution space easily, applying a balanced trade-off between intensification and diversification (see Section 3.1.2.9).

All LSN algorithms are based on the observation that searching a large neighbourhood results in finding local optima of high quality, where the neighborhood is defined implicitly by a destroy and a repair method (see Algorithm 14). We combined the general framework of LNS with an MILP solver in order to deal with the particularities of the supply chain problem

under study. Further details of our approach are provided in section 3.2.1.1.

3.1.2.8.2 Component Exchange Among Metaheuristics One of the most popular ways of metaheuristic hybridization is the use of trajectory methods inside population-based methods. Indeed, most of the successful applications of EC and ACO make use of local search procedures. The reason for that becomes apparent when analyzing the respective strengths of trajectory methods and population-based methods.

The power of population-based methods is certainly their capability of recombining solutions to obtain new ones. In EC algorithms explicit recombinations are implemented by one or more recombination operators. In ACO, for example, recombination is implicit, because new solutions are generated by using a probability distribution over the search space which is a function of earlier populations. This enables the search process to perform a guided sampling of the search space, usually resulting in a coarse grained exploration. Therefore, these techniques can effectively find promising areas of the search space.

The strength of trajectory methods is mainly the way in which they explore a promising region of the search space. In those methods local search is the driving component. Because of this, promising areas in the search space are searched in a more structured way than in population-based methods. Therefore, the danger of being close to good solutions but "missing" them is not as high as in population-based methods. More formally, local

search techniques efficiently drive the search toward the attractors, i.e., local optima or confined areas of the space in which many local optima are condensed.

In summary, population-based methods are better in identifying promising areas in the search space from which trajectory methods can quickly reach good local minim. Therefore, hybrid metaheuristics that can effectively combine the strengths of both population-based methods and trajectory methods are often very successful.

3.1.2.8.3 Integration of metaheuristics with artificial intelligence and operations research techniques One of the most prominent research directions is the integration of metaheuristics with more classical artificial intelligence (AI) and operations research (OR) methods, such as constraint programming (CP) and branch & bound as well as other tree search techniques. In the following we outline some of the possible ways of integration.

Metaheuristics and tree search methods can be sequentially applied or they can also be interleaved. For instance, a tree search method can be applied to generate a partial solution which will then be completed by a metaheuristic approach. Alternatively, metaheuristics can be applied to improve a solution generated by a tree-search method.

CP techniques can be used to reduce the search space or the neighborhood to be explored by a local search method. In CP, combinatorial

optimization problems are modelled by means of variables, domains and constraints, which can be mathematical (as for example in linear programming) or symbolic. Constraints encapsulate well-defined parts of the problem into sub-problems, thus making it possible to design specialized solving algorithms for sub-problems that occur frequently. Every constraint is associated to a *filtering* algorithm that deletes values from a variable domain that do not contribute to feasible solutions. Metaheuristics (especially trajectory methods) may use CP to efficiently explore the neighborhood of the current solution, instead of simply enumerating the neighbors or randomly sampling the neighborhood. A prominent example of such a kind of integration is Large Neighborhood Search (Shaw, 1998), which is the technique developed in this thesis in the context of integer programming. These approaches are effective mainly when the neighborhood to explore is very large, or when problems (such as many real-world problems) have additional constraints (usually called *side constraints*). A detailed overview of the possible ways of integration of CP and metaheuristics can be found in (Focacci et al., 2002).

Another possible combination consists in introducing concepts or strategies from either class of algorithms into the other. For example, the concepts of tabu list and aspiration criteria—known from tabu search—can be used to manage the list of open nodes (i.e., the ones whose child nodes are not yet explored) in a tree search algorithm. An example of such an approach can be found in (Della Croce and T'kindt, 2002). Tree-based search is also successfully integrated into ACO in (Blum, 2005), where beam search (Ow and Morton, 1988) is used for solution construction.

Integer and linear programming can be also effectively combined with metaheuristics. For instance, linear programming is often used either to solve a sub-problem or to provide dual information to a metaheuristic in order to select the most promising candidate solution or solution component (Ibaraki and Nakamura, 2006; Blum, 2005).

The kinds of integration we shortly mentioned belong to the class of *integrative combinations*. The other possible way of integration, called either *collaborative combinations* or also *cooperative search* consists in a loose form of hybridization, where search is performed by possibly different algorithms that exchange information about states, models, entire sub-problems, solutions or search space characteristics. Typically, cooperative search algorithms consist of the parallel execution of search algorithms with a varying level of communication. The algorithms can be different or they can be instances of the same algorithm working on different models or running with different parameter settings. The algorithms composing a cooperative search system can be all approximate, all complete, or a mix of approximate and complete approaches. This area of research shares many issues with the design of parallel algorithms and we forward the interested reader to the specific literature on the subject (Alba, 2005).

3.1.2.9 Intensification and Diversification

As mentioned before, (hybrid) metaheuristics are intelligent strategies for exploring a search space. Crucial for the success of such an algorithm is a well-adjusted (dynamic) balance between *diversification* and *intensifica-*

tion. The term diversification generally refers to the exploration of the search space, while the term intensification refers to the exploitation of the accumulated search experience. The balance between diversification and intensification is important because, first, we would like to quickly identify areas of search space with high quality solutions, and second, we would like to avoid spending too much time in areas of the search space that are already well explored or that only consist of poor-quality solutions. The interested reader is referred to Blum (2004) for further details on this topic.

3.2 Proposed methodology

In this section we describe in detail the methods developed in this Thesis to tackle the problems described above. We start by introducing a customized LNS method suitable for complex SCM problems. We then describe in detail an objective reduction method for MOO problems, with particular emphasis on those that incorporate several environmental objectives. We conclude the section describing an enhanced version of the epsilon constraint method (suitable for MOO problems) that incorporates objective reduction methods and random sequences.

3.2.1 Large neighborhood search applied to SCM

3.2.1.1 Large neighborhood search

The standard large neighborhood search (see Algorithm 14) was first introduced by (Shaw, 1998), the interested reader can find an excellent review

in (Gendreau and Potvin, 2010). The LNS belongs to a class of algorithms known as very large scale neighbourhood search (VLSN) (Ahuja et al., 2002; Blum C. Puchinger et al., 2011; Talbi, 2002).

In the LNS hybrid metaheuristic the neighborhood is defined implicitly by a destroy and a repair method. The destroy method removes part of the current solution while the repair method rebuilds the destroyed solution. The destroy method typically contains a random element, such that different parts of the solution are destroyed in every iteration of the method. The neighborhood $N(x)$ (see section 3.2.1.2) of a solution x is then defined as the set of solutions which are reached by first applying the destroy method and then the repair method. The main idea behind the LNS hybrid metaheuristic is that the large neighborhood allows the algorithm to explore the solution space easily, applying a balanced trade-off between intensification and diversification.

In what follows, the algorithm pseudo-code (see Algorithm 14) is presented in more detail. Three variables are kept by this method. The variable x^* is the best solution obtained during the search, x is the current solution, and x' is the promising solution that can be discarded during the current iteration. The function $destroy(.)$ is the destroy method while $repair(.)$ is the repair method. More specifically, $destroy(x)$ returns a copy of x that is partly destroyed. Applying $repair(.)$ to a partly destroyed solution repairs it, that is, it returns a feasible solution built from the destroyed one. In line 1 the global best solution is initialized. In line 3 the method applies the destroy method and then the repair method to obtain a new

solution x' . In line 4 the new solution is evaluated in order to determine whether this solution should become the new current solution or whether it should be rejected. The acceptance function can be implemented in different ways. Line 7 checks whether the new solution is better than the best known solution through the $objectiveValue(.)$ function. The best solution is updated in line 8 if necessary. In line 10 the termination condition is checked. Classical termination criteria are the limit on the number of iterations or the time limit.

Algorithm 14 The framework of LNS

Require: x feasible solution
Ensure: x^* local optima

1: $x^* := x$
2: **repeat**
3: $\quad x^{'} := repair(destroy(x))$
4: $\quad$ **if** $accept?(x', x^{)}$ **then**
5: $\quad\quad x := x'$
6: $\quad$ **end if**
7: $\quad$ **if** $objectiveValue(x') < objectiveValue(x^*)$ **then**
8: $\quad\quad x^* := x'$
9: $\quad$ **end if**
10: **until** stopping conditions

With regard to the acceptance criteria of solution x, there are three main alternatives: accepting only improved solutions, or applying acceptance criteria like in simulated annealing, or accepting randomly.

The destroy method is the most important part of the LNS method.

The most critical decision when implementing the destroy method is the degree of destruction: (Shaw, 1998) proposed to gradually increase the degree of destruction (see section 3.2.1.2), while Ropke and Pisinger (2006) choose the degree of destruction randomly in each iteration by choosing the degree from a specific range dependent on the instance size. The destroy method must also be chosen such that the entire search space can be reached, or at least the interesting part of the search space where the global optimum is expected to be found.

The repair method should be optimal in the sense that the best possible full solution is constructed from the partial solution. Alernatively, an heuristic approach can also be employed if one is satisfied with a good solution constructed from the partial solution.

3.2.1.2 Neighbourhood search space

The LNS decomposes the original problem into a number of smaller subproblems that are solved sequentially. Each sub-problem emerges from the current solution x, which is destroyed thereby giving rise to a partial solution x^p. The partial solution x^p defines a neighbourhood of solutions $N(x^p)$ (see Figure C.19) that can be explored rather fast by either tailored (e.g., another heuristic or meta-heuristic) or general purpose algorithms (e.g., branch and cut MIP solvers).

A neighbourhood is a function $N : F \rightarrow 2^F$ (see Figure C.19) that assigns to each $x'' \in F$ a set of neighbouring solutions $N(x^p) \subseteq F$, where

F is the space of all feasible solutions. Hereby, each solution $x'' \in N(x)$ is obtained by applying an operator to x^p, which performs a rather small change to x^p and therefore creates a new solution x''. Operators or movements are applied in a particular order.

To enable the control of the neighborhood size $|N(x^p)|$, initially, the x solution is destroyed slightly ($n = 1$). If during the search (repair(.)) a number m_{max} of consecutive attempted has not resulted in an improved solution, then the destruction process is increased by 1 up to n_{max}. Hence, there is an upper limit of n_{max}, which establishes the size of the neighborhood.

3.2.1.3 Solution Approach

LNS is a general framework that must be adapted to the particularities of the problem under study. Hence, the definition of the large neighbourhood is highly dependent on the problem of interest.

In our cases of study a solution is a set of integer decision variables. The the *destroy*(.) method consists of fixing an appropriate portion of the decision variables to the values that they have in the current solution x. The remaining "free" variables are then the only ones considered by the optimization algorithm.

The *repair*(.) method is implemented by an MILP-solver, which finds an improved solution, which might become the new current solution. A new large neighborhood is then defined around it, and the process is repeated in subsequent iterations.

Obviously, the selection of the decision variables that remain fixed and the ones that are subject to optimization, respectively, plays a crucial role in the performance of the algorithm. Particularly, the number of free variables directly defines the size of the neighbourhood. Too restricted neighbourhoods—that is, sub-problems—are unlikely to yield improved solutions, while too large neighbourhoods might result in excessive running times for solving the sub-problems by the MILPsolver. In our case we have adopted a strategy for dynamically adapting the number of free variables (see section 3.2.1.2), which are randomly selected.

The description of LNS implementation to deal with problems SC1 (section 6.7.1) and SC2 (section 6.7.3) follows:

The algorithm requires the following input data:

- A maximum execution time (t_{max})
- A maximum number of iterations (it_{max})
- A maximum number of variables to be released (n_{max}).
- A maximum number of attempts (m_{max}).

The algorithm works as follows (see Algorithm 15 or 17 and Figures C.20 and C.21). We generate first the initial solution and calculate the corresponding objective function value. The initial solution is a feasible solution with all the variables fixed. In the main loop, while *end* not equal

to true, for each trial m, we do the following. First, we randomly choose a set of n variables V to release. Second, we copy the solution s to s' and release the n variables from V. Third, we invoke the solver. The solver seeks to improve the solution changing the value of the variables released. The solve invocation gets the run time t employed by the solver, the new value of the objective function f_o and a new solution s''. If the objective function value is better than the current value, then we update the best solution, and the objective function, and assign to variable *improved* the value of true. Finally, we increase the number of iterations it, and the current time ct, until at least one of them exceeds the predefined limits. When this happens, then the variable *end* is assigned a value of true and the algorithm terminates. The detailed procedures used in our algorithm are described later in the case study section 4.

3.2.2 Objective reduction

Multi-objective optimization (MOO) is widely used in many areas of science and engineering for simultaneously optimizing several objective functions subject to some equality and inequality constraints. MOO has gained wider interest, being nowadays increasingly used in process systems engineering (PSE) (see section 7.2). Unfortunately, the complexity of MOO grows rapidly in size with the number of objectives (Deb and Saxena, 2005), requiring prohibitive computational times for medium/large size problems accounting for several objectives. Dimensionality reduction techniques aim at overcoming these limitations by identifying redundant objectives that can be omitted while still preserving the problem structure to the extent

Algorithm 15 LNS for Supply Chain

Require: model mdl, $t_{max} > 0$, $it_{max} > 0$, $m_{max} > 0$, $n_{max} > 0$
Ensure: solution s

1: $\langle s, fo \rangle :=$ initial_solution(mdl);
2: $end :=$ **FALSE** ; $it := 0$; $ct := 0$
3: **repeat**
4: $n := 1$; $improved :=$ **FALSE**;
5: **repeat**
6: $m := 1$;
7: **repeat**
8: $V := choose(n)$
9: $s' := release(s, V)$;
10: $\langle t, fo'', s'' \rangle := MILP(mdl, s')$;
11: **if** $better(fo, fo'')$ **then**
12: $fo := fo''$
13: $s := s''$
14: $improved :=$ **TRUE**
15: **end if**
16: $ct := ct + t$
17: $it := it + 1$
18: **if** $ct \geq t_{max}$ **OR** $it \geq it_{max}$ **then**
19: $end :=$ **TRUE**
20: **end if**
21: $n := n + 1$;
22: **until** $m > m_{max}$ **AND** $improved$
23: $m := m + 1$;
24: **until** $n > n_{max}$ **AND** $improved$
25: **until** end

possible. Objective reduction methods facilitate in turn the post-optimal analysis of the Pareto solutions, since they lower the number of objectives to be considered in the decision-making process.

3.2.2.1 Background and Model

In this subsection we extend the definitions presented in section 7.3. We follow the work by Brockhoff and Zitzler (2006a), in which the reader will find further details.

Let $MO(x)$ be a multi-objective minimization problem of the following form:

$$MO(X) = \min_x \{f(x) := (f_1(x), ..., f_k(x)) : x \in X\} \tag{3.16}$$

with k objective functions $f_i := X \longrightarrow \mathbb{R}$, $1 \leq i \leq k$, where each objective function f_i maps a solution $x \in X$ to a value of the function vector $f := (f_1, ..., f_k)$. The next 4 definitions are fundamental to understand how objective reduction algorithms work:

Definition The weak Pareto dominance relationship is defined as follows: $\preceq_{F'} := \{(x, y) \mid x, y \in X \wedge \forall f_i \in F' : f_i(x) \leq f_i(y)\}$, where F' is a set of objectives with $F' \subseteq F := \{f_1, \ldots, f_k\}$.

Therefore:

Definition x weakly dominates y ($x \preceq_{F'} y$) with respect to the objective set F' if $(x, y) \in \preceq_{F'}$.

and

Definition $x^* \in X$ is called Pareto optimal if there is no other $x \in X$ that dominates x^* with respect to the set of all of the objectives.

Finally,

Definition Two solutions x, y are non-dominated or incompatible if neither weakly dominates the other one.

We present an illustrative example in section 7.3.1 to clarify the concepts presented before.

The notion of conflicts between objectives can be generalized by introducing the concept of δ error (Zitzler et al., 2003). In the following paragraphs, we formalize this definition in the following paragraphs:

Definition Let F_1 and F_2 be two objective sets. We define
$F_1 \sqsubseteq^{\delta} F_2 :\Longleftrightarrow \preceq_{F_1} \subseteq \preceq^{\delta}_{F_2}$.

Definition Let F_1 and F_2 two objective sets. We call

- F_1 δ-nonconflicting with F_2 iff $F_1 \sqsubseteq^{\delta} F_2$ and $F_2 \sqsubseteq^{\delta} F_1$.
- F_1 δ-conflicting with F_2 iff not δ-nonconflicting with F_2.
- Note that $\preceq^{\epsilon}_{F'} := \{(x, y) \mid x, y \in X; \forall i \in F' \subseteq F : f_i(x) - \epsilon \leq f_i(y)\}$.

Definition Let F be a set of objectives and $\delta \in \mathbb{R}$. An objective set $F' \subseteq F$ is denoted as:

- δ-minimal wrt F iff:
  - F' is δ-nonconflicting with F,
  - F' is δ'-conflicting with F, $\forall\ \delta' < \delta$,
  - there exists no $F'' \subset F'$ that is δ-nonconflicting with F.
- δ-minimum wrt F iff:
  - F' is δ-minimal wrt F,
  - there exists no $F'' \subset F$ with $|F''| < |F'|$ that is δ-minimal wrt F.

Definition A set F of objectives is called δ-redundant iff there exists $F' \subset F$ that is δ-minimal wrt F.

The problem called "minimum objective subset (MOSS)", proposed in Brockhoff and Zitzler (2006c), can be generalized to give rise to the δ-MOSS problem by allowing an error δ:

Definition Given a set of Pareto solutions with F objectives and $\delta \in \mathbb{R}$ error, the problem δ-MOSS consists of computing the δ-minimum objective subset $F' \subset F$ wrt F.

Definition Given a set of Pareto solutions with F objectives and $k \in \mathbb{N}$, the problem k-MOSS solved by computing the objective subset $F' \subset F$ which has size $|F'| \leq k$ and is δ-nonconflicting with F with the minimal possible δ.

In this work, we present an algorithm to reduce the dimension (i.e., number of objectives) of MOO problems. According to the definitions presented before and given a set of objectives $F := (f_1, \ldots, f_k)$, our goal is to determine a subset F' of F ($F' \subseteq F$) of given cardinality such that the error δ of removing the objectives not included in F' is δ-minimum. To this end, we have developed a method based on an MILP approach presented before by the authors (see section 7.4).

3.2.2.2 Solution strategy for objective reduction

The pivotal idea of our method is to decompose the original full space MILP for dimensionality reduction into two sub-problems (see section 7.5): a lower level and an upper level sub-problem in order to solve faster than the full-space MILP. Our method make use of a set of cutting planes in order to tighten the relaxation of the upper bounding sub-problem, thereby expediting its solution. A brief explanation the algorithm is given below (further details are available in section 7.5 and 3.2.2.5):

The proposed decomposition method (see Figures C.8, C.9 Algorithms 16 and 20) solves in each iteration a lower level problem MOR_{μ_p} and an upper level problem until the difference between the lower and upper bounds provided by these two sub-problems falls within an epsilon tolerance ϵ previously defined in the Algorithm 16).

The lower level problem is very similar to the full-space MILP, but only defined for a subset of size μ of Pareto solutions. Because of the way in which they are constructed, the lower bounding MILPs contain fewer binary variables than the original MILP. Furthermore, in each lower level problem

we add a set of cutting planes obtained from previous sub-problems of size μ' , where $\mu' < \mu$. The lower level problem produces a solution entailing a set of objectives identified that should be removed from the origianl MOO model. This solution is used by the upper bounding problem to expedite its solution (see section 3.2.2.3).

The upper bounding problem consists of a customized algorithm that calculates the delta error faster than the full space MILP by using the solution provided by the lower level problem (see section 3.2.2.4).

3.2.2.3 Lower level

The lower level problem corresponds to the original MILP that is defined only for a subset of solutions of small size (defined by variable μ). To accelerate the solution of these sub-problems, we add cutting planes that are obtained from previous sub-problems already solved (see Equations 7.12-7.15). The solution obtained from each sub-problem MOR_{μ_p} has the property of providing a lower bound on the global optimum of the full space MILP. This property allows approximating the solution of the full space MILP without having to solve it explicitly for problems involving a very large number of solutions and objectives (see proof in Appendix B.1).

3.2.2.4 Upper level

The upper level problem computes the delta error for the combination of objectives predicted by the lower bounding MILP. In this level we apply a customized algorithm that calculates the delta error for a given combination

of objectives. This method is described in detail in Algorithm 17.

3.2.2.5 Algorithmic Steps

The detailed steps of the proposed decomposition strategy are as follows (see Algorithm 16):

The algorithm requires the following input data:

- A maximum execution time (t_{max}).
- A maximum number of iterations (it_{max}).
- A maximum size buffer for storing cuts (λ).
- Initial problem size (μ_0).
- An optimal criteria (ϵ).

1. Set upper bound $UB := +\infty$, lower bound $LB := -\infty$, problem size $\mu := \mu_0$ (see Line 2 of Algorithm 16).
2. Outer loop: while *end* condition is not satisfied (Lines 3 and 16-18, 19-21 and 24-16), otherwise algorithm stops.
   (a) Inner loop: while *end* condition is not satisfied (Lines 5 and 16-18, 19-21 and 24-16) and problem counter is not exceeded ($p \leq \frac{N}{\mu}$), otherwise if *end* condition is not satisfied set $p := 1$ (Line 4) and $\mu := r * \mu$ (Line 23) and Go to step 2.

(b) Solve lower bounding problem MOR_{μ_p} pointed out by p with size μ using cutting planes in list Cut_{LB}. A lower bound on the delta value (δ_{LB}) and a set of objectives to be removed are obtained $ZO(i)$ (Line 6).

(c) Solve upper bounding problem fixing binary variables $ZO(i)$ and obtain an upper bound on the delta value(δ_{UB}) (Line 7).

(d) The list of cuts Cut_{LB} is updated: A new cut δ_{LB} is then inserted in the corresponding position in the list of cuts Cut_{LB}, replacing a previous cut when the list is filled. The list Cut_{LB} is sorted in descending order of delta values for all of the lower bounding problems solved in previous iterations (Line 14).

(e) Increase loop control variables: problem counter p, iterations it and time t (Line 15) and Go to step 2.

(f) The LB and UB are updated: if $LB < \delta_{LB}$ then $LB := \delta_{LB}$ and if $UB > \delta_{UB}$ then $UB := \delta_{UB}$ (Lines 16-21).

The method exposed above presents some remarks:

- The lower bounding MILPs can be either solved to global optimality or stopped when an optimality gap is reached. When the second option is selected (which expedites the solution of the sub MILPs), the cutting planes are constructed considering the best possible bound obtained by the branch and cut algorithm (instead of the delta value of the best integer solution).

- Note that we have slightly modified the MILP introduced by Guillén-Gosálbez (2011a) in order to calculate the approximation error in the

Algorithm 16 Divide & Conquer

Require: $\lambda > 0$, $N \geq \mu_0 > 0$, $\mu_0 \mid N$, $t_{max} > 0$, $it_{max} > 0$, $\epsilon > 0$
Ensure: : LB is the minimum δ error of the ps.

function $ComputeDeltaError(mdl : Model,\ ps : Matrix[N, K]$ of $\mathbb{R},$ $\lambda : \mathbb{N},\ \mu_0 : \mathbb{N},\ t_{max} : \mathbb{N},\ it_{max} : \mathbb{N},\ \epsilon : \mathbb{R})$
  $UB := \infty$; $LB := -\infty$; $\mu := \mu_0$;
  while $NOT\ end$ **do**
    $p := 1$;
    while $NOT\ end\ AND\ p \leq \frac{N}{\mu}$ **do**
      $\langle \delta_{LB}, ZO(i), t_1 \rangle := SolveMOR(mdl, ps, Cut_{LB}, p)$;
      $\langle \delta_{UB}, t_2 \rangle := customizedDeltaError(mdl, ps, ZO(i))$;
      if $UB > \delta_{UB}$ **then**
        $UB := \delta_{UB}$
      end if
      if $LB < \delta_{LB}$ **then**
        $LB := \delta_{LB}$
      end if
      $insertSorted(Cut_{LB}, \langle \delta_{LB}, p \rangle)$;
      $p := p + 1$; $it := it + 1$; $t := t + t_1 + t_2$;
      if $UB - LB \leq \epsilon$ **then**
        $end = TRUE$;
      end if
      if $t > t_{max}$ OR $it > it_{max}$ **then**
        $end := TRUE$;
      end if
    end while
    $\mu := \mu * r$;
    if $\mu > N$ **then**
      $end := TRUE$;
    end if
  end while
end function

same manner as proposed by Brockhoff and Zitzler (2006a). Hence, to determine the delta error, we consider any pair of solutions such that one dominates the other in the reduced space regardless of whether both solutions are Pareto optimal in such a reduced domain. Figure C.10 provides an illustrative example on this issue.

3.2.3 Enhanced ϵ-constraint method

The ϵ-constraint method divides the domain of each objective transferred to an auxiliary constraint into equal intervals, and then solves an exponential number of single-objective problems resulting from making all possible combinations of objectives values. In general, by choosing p partitions for each objective, we solve $k + (p-1)^{k-1}$ single-objective problems for the case of k objectives, (note that we first need to optimize in turn each single objective separately). Hence, the complexity of this method grows exponentially in size with the number of objectives, and can get out of hand very easily for problems with several objectives (see section 8.1). Besides, the use of an arbitrary of selecting the required epsilon parameter can lead to an important waste of computational effort resulting from attempts to solve instances which are either infeasible or which produce repeated solutions. In this work we introduce an enhanced ϵ-constraint method that integrates two main techniques:

1. A rigorous objective reduction technique that eliminates redundant objectives.

2. Pseudo/quasi-random sequences that allow generating in a more efficient manner the values of the ϵ parameters. We explain in detail

both ingredients of the method in the ensuing sections.

3.2.3.1 Dimensionality reduction

Dimensionality reduction methods allow eliminating redundant objectives in *MOO* problems (see Figure C.14). Consider the general *MOO* model introduced in sections 3.1.1.6 and 8.2 . The goal of dimensionality reduction is to find a subset F' of objectives functions pertaining to the original set F with the following property: when we optimize the problem in the reduced domain of objectives F' instead of the original domain F, we will generate in less CPU time a Pareto front that is very close to the Pareto front of the original problem. A review about dimensionality reduction can be found in sections 7.2 and 8.3.1. In this work, we use the dimensionality reduction method introduced in section 3.2.2 and in chapter 7 to expedite the performance of the ϵ-constraint method.

3.2.3.2 Random sequences

During the application of the ϵ-constrain method, we partition the domain of each objective into equal intervals. This leads to large number of single-objective models, most of which might be either unfeasible or produce the same Pareto solution. To overcome this limitation to the extent possible, we propose to generate the ϵ parameters using pseudo/quasi-random sequences. Particularly, in this work we focus on the following strategies: the pseudo-random sequence (uniform distribution), and the Halton and Sobol sequences (quasi-random).

The pseudo-random sequence is well known, and therefore we will not get deeper into its details. The Halton and Sobol sequences (also called low-discrepancy or quasi-random sequences) are useful in numerical integration, as well as in simulation and optimization. Surveys of applications of low-discrepancy sequences can be found in Fox (1986); Bratley and Fox (1988); Bratley et al. (1992).

These sequences are used, in the context of our work, to generate the values of the ϵ parameters in a more effective manner. The goal is to spread these values as much as possible so they cover a wider region of the search space, which will eventually lead to a better Pareto front.

The concept of *low-discrepancy* is introduced in section 8.3.2 and measures the error in hypervolume estimation (see Figure C.15).

3.2.3.2.1 Halton sequence The Halton sequence, which was introduced by Halton (1960), can be used like a random number generator to produce points in the interval $[0, 1]$. More details on this strategy can be found in section 8.3.2.1.

3.2.3.2.2 Sobol sequence The Sobol sequence is the other low-discrepancy sequence applied in this work in order to generate random numbers. This sequence was introduced by Sobol (1967). We give an informal description in section 8.3.2.2.

3.2.3.3 Detailed steps of the algorithm

The algorithm proposed is this thesis is presented in detail in section 8.3.2.3 (and see Algorithm 21), and here we just summarize the main aspects:

- The first step is to generate an initial set of Pareto points on the basis of which we will perform the objective-reduction analysis. To this end, we can apply a heuristic approach consisting of solving a set of bi-criteria problems in which we trade-off one objective (when dealing with environmental problems, we will choose the economic performance) against each of the remaining criteria (i.e. the individual environmental indicators). For each of these bi-criteria problems, the ϵ-constraint method is executed for a given number of iterations, producing a set of Pareto solutions. The objective reduction method is then applied to identify and eliminate redundant objectives (Step 1 of Algorithm 21).

- A set of values of the epsilon parameters is next generated using the pseudo/quasi-random sequences. The single-objective problems are finally solved for these parameters values (Step 2 of Algorithm 21).

Chapter 4

Results

In this section we provide numerical results obtained from the application of the algorithms described above.

4.1 Large neighborhood search: generalities and tuning of the algorithm

We apply the LNS algorithm described above to solve a spatially explicit SCM problem. The goal is to determine the structure of a SC (in this case, a three-echelon SC: production-storage-market, as shown see Figure C.1). This network includes a set of production and storage facilities, where products are stored before being delivered to the final customers. The facilities can be installed in a set of sub-regions that correspond to the potential locations in which the overall regions of interest is divided. The different SC problems to be solved are summarized in the following paragraphs:

- **Case SC1:** single-objective optimization of the hydrogen supply chain (see section 6.7.1): given are a fixed time horizon and number of time periods, the set of available production, storage and trans-

portation technologies, the capacity limitations of plants and storage facilities, the costs associated with the network operation, the investment cost, and the interest rate. The final goal is to determine the SC design, including the number, type, location and capacity of plants and storage facilities; the number and type of transportation units and transportation links to be established between the potential locations; and the associated planning decisions, including the production rates at the plants, the inventory levels at the storage facilities and the hydrogen flows between plants and storage facilities; in order to minimize the total cost.

- **Case SC2:** single-objective optimization of ethanol supply chains (see section 6.7.3): given are a fixed time horizon, product prices, cost parameters for production, storage and transportation of materials, demand forecast, tax rate, capacity data for plants, storages and transportation links, fixed capital investment data, interest rate, storage holding period and landfill tax. The goal is to determine the configuration of a three-echelon bioethanol supply chain and the associated planning decisions with the objective of maximizing the economic performance calculated over the entire useful life of the SC.

Cases SC1 and SC2 (see section 2) are both related with energy applications that have attracted an increasing interest in recent years. Both cases are taken as benchmark to test the capabilities of our algorithm. For each case study, we provide a general definition of the problem, and then discus some implementation (see sections 6.6 and 3.2.1.1) details of the algorithm before presenting the numerical results (see section 6.7).

In the following subsections we present numerical results that illustrate the performance of our solution method compared with the commercial full space branch and cut code implemented in CPLEX, where the numerical experiments were performed on a PC Intel (R) Core (TM) Quad CPU Q9550@2.83 GHz and 3 GB of RAM.

We have selected different instances of the models comprising 2, 4, 6, 8, 10, 12, 14 and 16 time periods, respectively. The MILP size is shown in Table C.1. In Table C.1, we display the number of continuous variables, the number of binary variables, the number of integer variables, the number of decision variables (Bin+Int), the total number of variables, and finally, the number of equations.

Firstly, we tune the algorithm in order to determine tight values of the input parameters m_{max} and n_{max} by solving problems for different values of n_{max} and m_{max}. Recall that n_{max} is the maximum number of variables released and m_{max} the maximum number of attempts. Figures C.2 and C.4 show the results obtained for 10 runs of the algorithm, where the final values are highlighted in red color.

We next provide further details on the tuning of the algorithm for the two case studies solved in this section:

- Case SC1 (section 6.7.1): design of hydrogen SCs for vehicle use (Sabio et al., 2010).

The first SC (SC1) design problem has as objective to determine the configuration of a three-echelon hydrogen network for vehicle use (production-storage-market) with the goal of minimizing the total cost (Sabio et al., 2010). More details can be found in sections 6 and A.1.

The model includes three main types of discrete variables that are relevant in the implementation of our algorithm (see sections 6 and A.1):

- **Integer variables** N^{PL}_{igpt}: Number of facilities producing hydrogen in form i using technology p established in location g at period t.
- **Integer variables** N^{ST}_{gst}: Number of storage facilities of type s opened in location g at period t.
- **Binary variables** $X_{gg'lt}$: Equal 1 if there is a link between g and g' using transportation mode l in period t and 0 otherwise.

- $initial\_solution(mdl)$: The initial solution in this case is generated by solving the hydrogen model (MH) with variables NP^{PL}_{igpt}, N^{ST}_{gst} and $X_{gg'lt}$ fixed to the values obtained from a reduced-space model that considers a single time period with a demand equal to the average value over all the time periods (see Algorithm 18).
- $choose(n)$: this function selects n places. In the case of the hydrogen SC, we consider a set of regions pertaining to Spain, which are numbered from 1 to 19 (see Figure C.22).

– $release(s, V)$: This method returns a copy of the solution s with some variables released. The variables are released as follows: For every variable PR_{igpt} or N^{ST}_{gst} or $X_{gg'lt}$ where $g \in v$ impose the lower bounds

$$s[PR_{igpt}].lb = LB$$

$$s[N^{ST}_{gst}].lb = LB$$

$$s[X_{gg'lt}].lb = 0$$

and impose the upper bounds

$$s[PR_{igpt}].ub = UB$$

$$s[N^{ST}_{gst}].ub = UB$$

$$s[X_{gg'lt}].ub = 1$$

.

– $MILP(mdl, s')$: For a model of type mdl MILP, and a solution s', this function solves the model. The solver tries to improve the solution changing the value of the variables released, while the function itself implements a branch & cut method. We used a commercial software called CPLEX that implements the state of the art branch & cut theory.

– $better(fo, fo')$: Given two objective functions fo y fo', this function identifies the best, which is the smallest one, since we are dealing with a minimization problem.

- Case SC2 (section 6.7.3): design supply chains for ethanol production (Kostin et al., 2011a). Application to the sugar cane industry of Argentina.

  The second SC (SC2) example addresses the design of supply chains for ethanol production. We consider a generic three-echelon SC (production-storage-market). The model is described in detail in (Kostin et al., 2011a) and it is summarized in section A.2.

  The most relevant variables manipulated by our algorithm are the following:

  - **Integer variables** NP_{pgt}: Number of factories of technology p established in region g at period t.
  - **Integer variables** NS_{sgt}: Number of storage facilities s opened in region g at period t.
  - **Binary variables** $X_{lgg't}$: Equals 1 if there is a link between g and g' using transportation mode l at period t and 0 otherwise.

  The initial solution for model ETHANOL is generated as follows. We first solve a relaxed MILP in which some integer variables are treated as continuous ones. The solution provided by this relaxed model is then rounded to the nearest integer, and an LP with fixed binaries is finally solved (see Algorithm 19).

  In the ethanol SC, the implementation of the methods is as follow:

  - $initial\_solution(mdl)$: we obtain the initial solution by solving

a relaxed problem where the integer variables are treated as continuous (see Algorithm 19).

- $choose(n)$: this function selects n places. In the case of ethanol, they are the provinces of Argentine, which are numbered from 1 to 25 (see Figure C.23).
- $release(s, V)$: this method returns a copy of the solution s with some variables released. The variables are released as follows: For every variable NP_{gpt} or NS_{sgt} or $X_{lgg't}$ where $g \in v$ impose the lower bounds

$$s[NP_{gpt}].lb = LB$$

$$s[NS_{sgt}].lb = LB$$

$$s[X_{lgg't}].lb = 0$$

and impose the upper bound

$$s[NP_{gpt}].ub = UB$$

$$s[NS_{sgt}].ub = UB$$

$$s[X_{lgg't}].ub = 1$$

.

- $MILP(mdl, s')$: is the same as for the hydrogen case. We used a commercial software called CPLEX to solve the MILP subproblems.
- $better(f_o, f'_o)$: given two objective functions fo y fo', this func-

tion determines the best of two. The best solution shows the greatest objective function values, since we are dealing with a maximization problem.

4.1.1 Numerical results

4.1.1.1 Case SC1: Design of hydrogen SCs for vehicle use

Case SC1 design problem has as objective to determine the configuration of a three-echelon hydrogen network for vehicle use (production-storage-market) with the goal of minimizing the total cost. The complete mathematical formulation for this problem can be found in Sabio et al. (2010) and it is summarized in the Appendix A.1.

The initial solution in this case is generated by solving the model with its integer variables fixed to the values obtained considering only a single time period with a demand equal to the average demand over all time periods (see Algorithm 18 and section 6.7.1).

Figure C.3 shows the evolution of the lower and upper bounds found by CPLEX along with the values provided by the proposed LNS algorithm. As can be seen, for low time periods (less than or equal to 6), CPLEX outperforms the proposed algorithm, finding better solutions in shorter CPU times. On the contrary, for more than 6 time periods, CPLEX cannot find any solution, whereas LNS is always able to provide at least one feasible solution. Note that the variability of the results obtained with our algorithm is rather low.

In Table C.2, we provide the time at which the best solution calculated by the LNS has been found, the best objective function value in all the runs, the average objective function value and CPU time, and the standard deviation of the objective function and CPU time. Note that in periods 14 and 16 the standard deviation of the CPU time is zero because the algorithm convergence is very fast.

Finally in Table C.3 the optimality gaps of the following solutions are displayed: the best solution calculated by CPLEX after 12 hours of CPU time and after the same CPU time provided to the LNS, the best solution found by the LNS and the average solution calculated by the LNS. Note that in some instances, CPLEX is unable to provide any bound even after the aforementioned CPU time.

4.1.1.2 Case SC2: Ethanol supply chain model

Case SC2 addresses the design of supply chains for ethanol production. We consider a generic three-echelon SC (production-storage-market). The model is described in detail in Kostin et al. (2011b) and it is summarized in the Appendix A.2.

The initial solution for model ETHANOL is generated as follows. We first solve a relaxed MILP in which some integer variables are treated as continuous ones. The solution provided by this relaxed model is then rounded to the nearest integer, and an LP with fixed binaries is finally solved (see Algorithm 19).

Figure C.5 shows that for a small number of time periods (less than 8), CPLEX outperforms the proposed algorithm. However, for 8 time periods our algorithm outperforms CPLEX. For more than 8 time periods, none of the methods performs better than the other one for the whole range of CPU times. Particularly, LNS tend to behave better for short CPU times, whereas CPLEX provides better solutions when long CPU times are considered. The Table C.5 is equivalent to Table C.2, whereas Table C.6 is equivalent to Table C.3. Finally, for this case we observed that LNS provides better solutions (i.e., with better optimality gaps) for problems with larger sizes containing 12, 14 and 16 time periods.

4.2 Objective reduction

We applied our objective reduction algorithm to two MOO problems (see section 2): the MOO of hydrogen supply chains (see section 7.6.1) and the MOO of metabolic pathways (se section 7.6.2).

- **Case SC3:** multi-objective optimization of the hydrogen supply chain (see section 7.6.1); the SC topology is similar to the one presented in the previous case, but several additional objectives were appended to the objective function in order to reformulate the original single-objective model into a MOO one. The new formulation attempts to optimize a set of environmental indicators that are quantified following the life-cycle methodology. Particularly, we use the

Eco-indicator-99 framework, which considers 15 individual impact metrics in three damage categories (see sections 7.6.1 and A.1).

- **Case ethanol metabolic**: multi-objective optimization of the synthesis rate of ethanol in the fermentation of *Saccharomyces cerevisiae* (see section 7.6.2). Given a metabolic network (see Figure C.13) described by a Genralized Mass Action (GMA) model, the goal is to determine a set containing the preferred enzymatic profiles that optimize the synthesis rate of a metabolite at minimum cost and minimum increase in the concentration of intermediate metabolites (Pozo and Guillén, 2012).

Case SC3 and the model of metabolic pathways, both presented in section 2. The case SC3 addresses the design of hydrogen supply chains (a problem in the area of green engineering), while the second model deals with the multi-objective optimization of metabolic networks (an important area in systems biology). In both cases, our method was compared against the full space MILP model introduced by Guillén-Gosálbez (2011a), and the exact and greedy methods developed by Brockhoff and Zitzler (2006a). All the numerical experiments were conducted on a computer Intel(R) Core (TM) i7-3612QM CPU@ 2.10GHz 2.10GHz and 6GB of memory RAM (see section 7.6).

4.2.1 Multi-objective optimization of hydrogen supply chains for vehicle use

In this example, we dealt with the optimal design of a hydrogen SC for vehicle use in Spain taking into account economic and environmental concerns (see section 7.6.1). The goal is to determine the optimal network configuration in terms of its economic and environmental performance. The problem can be formulated as a multi-objective MILP that seeks to minimize the total cost of the network and its environmental impact. The environmental impact was calculated through 15 life cycle assessment indicators based on the Eco-indicator 99 methodology. Further details on this case study can be found in Sabio et al. (2010).

Our dimensionality reduction method was applied to 300 normalized Pareto solutions obtained from this model. The normalization procedure can be found in Appendix B.2. Particularly, we solved different instances of the dimensionality reduction problem, in each of which the goal was to eliminate a given number of objectives ranging from 1 to 15. The MILP for dimensionality reduction contains 4485034 variables and 7445102 equations.

Table C.7 shows that the MILP and our approach solve the problem to global optimality (i.e., with an optimality gap of 0%), whereas the greedy algorithm provides no information regarding the gap of the final solution found. As observed, our method identifies the optimal solution faster than both the MILP (Guillén-Gosálbez, 2011a) and the exhaustive method proposed by Brockhoff and Zitzler (Brockhoff and Zitzler, 2006a) in almost all

of the cases solved. Particularly, our method lead to reductions of CPU time 1 of up to 3 orders of magnitude when compared to the other methods (i.e., the full space MILP and the exhaustive method, while still guaranteeing the global optimality of the solution found).

Although the greedy algorithm is always the fastest approach, it offers no theoretical guarantee of reaching the global optimum. This high performance of the greedy algorithm is mainly due to the simple search strategy it implements, which starts with a reduced set of one objective, and then adds progressively the objective that leads to the minimum error, and keeps on doing this until the desired number of objectives kept is reached. This strategy performs very well in practice, but offers no guarantee of convergence to the global optimum.

Note also that the exhaustive method is the best algorithm only for those cases with a small number of objectives kept (i.e., when removing more than 12). The exhaustive method calculates the error for every possible combination of objectives kept. When the number of objectives kept is small, this strategy performs well because the number of potential combinations is small. On the contrary, for a large number of potential combinations, the exhaustive algorithm will very likely perform worse than our method.

4.2.2 Multi-objective optimization of metabolic networks

The use of mathematical optimization to improve biotechnological processes has the potential to produce significant economical savings, because of the

reduction in the number of experiments required to find strains with improved features. Moreover, the optimization procedure could provide valuable insight into the behavior of biological systems, thereby enhancing our understanding of cellular metabolism.

Given a metabolic network (see Figure C.13) described by a Generalized Mass Action (GMA) model, the goal is to determine a set containing the preferred enzymatic profiles that optimize the synthesis rate of a metabolite at minimum cost (minimum number of changes in the enzyme activities, i.e. minimum change in gene expression) and minimum increase in the concentration of intermediate metabolites in the fermentation of *Saccharomyces cerevisiae* for ethanol production. We consider a total of 15 objectives (Pozo and Guillén, 2012).

- Objectives $1-8$ correspond to changes in the enzyme expressions that should be minimized so as to make it easier to manipulate genetically the strain of interest and maintain cells homeostasis (enzymes K1-K8). K1: Hexose transporters, K2: Glucokinase/Hexokinase, K3: Phosphofructokinase, K4: Trehalose 6-phosphate syntase complex (+Glycogen production), K5: Glyceraldehyde-3-phosphate dehydrogenase, K6: GOL (Glycerol production), K7: Pyruvate kynase, K8: ATPase.
- Objective 9 is the total cost (the cost of changing the enzyme activities).
- Objective 10 is the synthesis rate of ethanol (that should be maxi-

mized).

- The remaining 5 objectives represent the concentration of metabolites X1-X5, which should be minimized and be kept as close as possible to those in the basal state to ensure the cell's homeostasis. X1: Internal glucose, X2: Glucose-6- phosphate, X3: Fructose-1,6-diphosphate, X4: Phosphoenolpyruvate, X5: Adenosine triphosphate (see section 7.6.2).

For this case, we generated 300 normalized Pareto solutions (see Appendix B.2) using the ϵ-constraint method (Haimes, Y.Y.; Lasdon, L.S.; Wismer, 1971). We then solved the dimensionality reduction problem, where the goal was to minimize the delta error associated with the elimination of a number of objectives ranging from 1 to 14. This case contains 1421528 variables and 3529528 equations.

Table C.8 shows the results obtained, in which the MILP and our approach are both solved with an optimality gap of 0%. As observed , our method identifies the optimal solution much faster than the MILP (Guillén-Gosálbez, 2011a) and the exhaustive method (Brockhoff and Zitzler, 2006a), obtaining reductions in the CPU time of approximately 3 orders of magnitude when compared to the other exhaustive methods that guarantee global optimality. The greedy algorithm is the fastest, but recall that it offers no theoretical guarantee of global optimality. In fact, for this case it provides solutions that are suboptimal.

The exhaustive method is the best algorithm only for the case with 1

objective kept (14 objectives removed). Note that the MILP method fails to provide a solution for those cases with a small number of objectives kept (i.e, more than 11 objectives removed).

4.3 Enhanced ϵ-constraint method

We test the capabilities of our method (see Algorithm 21 and section 3.2.3) through its application to cases SC3 and SC4 (supply chain design problems), which optimize the economic performance along with a set of life cycle assessment metrics. We compare in both cases our algorithm with the standard ϵ-constraint. These problems are presented in sections 8.4.1 and 8.4.2. Further details can be found in Kostin et al. (2012) and Sabio et al. (2010) .

Particularly, the following approaches are tested in the case studies (see Algorithm 22):

- Standard ϵ-constraint algorithm (equidistant ϵ values).
- Standard ϵ-constraint algorithm with pseudo-random ϵ values.
- Standard ϵ-constraint algorithm with Halton ϵ values.
- Standard ϵ-constraint algorithm with Sobol ϵ values.
- Enhanced ϵ-constraint algorithm with equidistant ϵ values
- Enhanced ϵ-constraint algorithm with pseudo-random ϵ values.
- Enhanced ϵ-constraint algorithm with Halton ϵ values.

- Enhanced ϵ-constraint algorithm with Sobol ϵ values.

In order to assess the experiments, two different indicators are used to quantify the quality of the Pareto front produced by each of these methods. The first is the number of unique Pareto solutions generated in a given time frame. The second is the quality of the Pareto front. The latter indicator is quantified using the hypervolume indicator. The hypervolume indicator is presented in section 8.4.

The multi-objective MILPs were implemented in GAMS, where the objective reduction algorithm was also coded. The MILPs were solved by CPLEX 12.5.1.0 on an Ubuntu 13.10 with an Intel i7-4770, 3.4GHz 8-cores processor, and 16GB of RAM. The Halton and Sobol sequences were implemented in GAMS as an extrinsic function. The objective reduction was carried out by the algorithm presented in section 3.2.2.

4.3.1 Sustainable planning of ethanol supply chain

- **Case SC4**: multi-objective optimization of the ethanol supply chain (see section 8.4.1): the SC topology is similar to the one presented in the previous case, but it was reformulated into a MOO by appending to the objective function 5 environmental indicators based on the Eco-indicator 99 framework (see sections 8.4.1 and A.2).

The first example (SC4) aims at determining the configuration of a three-echelon bioethanol network and associated planning decisions that maximize the net present value and minimize the environmental impact (Kostin et al., 2012).

Following the comparison process (explained in section 8.4.1), we obtained the results depicted in Figure C.17. As observed, in terms of number of feasible solutions generated and hypervolume value, the best approach is our ϵ-constraint method.

We found that among the pseudo/quasi-random sequences, the one with better performance is the Halton sequence. Furthermore, numerical results reveal that objective reduction techniques improve the performance of the algorithm from the viewpoints of number of feasible solutions generated and quality of the Pareto front (see section 8.4.1 and Figure C.17).

4.3.2 Sustainable planning of hydrogen supply chains

- **Case SC3**: multi-objective optimization of the hydrogen supply chain (see section 7.6.1). The SC topology is similar to the one presented in the previous case, but it was reformulated into an MOO by appending to the objective function 15 individual indicators quantified according to the Eco-indicator 99 framework (see sections 7.6.1 and A.1).

According to our comparison process explained in section 8.4.2, we obtained the results depicted in Figure C.18. As seen, our approach requires less CPU time to find feasible solutions. The standard ϵ-constraint (equidistant in the full space) is unable to obtain feasible solutions in 300 iterations, since for this case we need to solve $(N-1)^{15}$ single-objective problems, most of which are unfeasible. Regarding the hypervolume indicator, we

observe that our approach is clearly better than the rest, mainly because it can generate a larger amount of feasible points that cover a wider range of the search space.

Finally, we found that among the pseudo/quasi-random sequences, the best one is the Sobol sequence. Hence, objective reduction improves the performance of the standalone ϵ-constraint method from the viewpoints of number of feasible solutions per time unit and quality of the Pareto front (see section 8.4.2 and Figure C.18).

Chapter 5

Conclusions & Future work

5.1 Conclusions

From the results presented above, we draw the set of conclusions listed below:

- An efficient hybrid metaheuristic algorithm has been developed for spatially explicit SCM models. Our algorithm consists of a hybrid metaheursitic which combines the metaheuristic LNS with the commercial available branch-and-cut software CPLEX. The capabilities of the proposed technique were illustrated through its application to two SCs that focus on energy applications: the design of SCs for sugar and ethanol production and the strategic planning of infrastructures for hydrogen production. Our approach proved to obtain near optimal solutions in a fraction of the time spent by the full space MILP. Furthermore, this method allows identifying feasible solutions even in those cases in which CPLEX fails to converge. Numerical examples demonstrate that our method is particularly suited for tackling large scale problems with high number of time periods and potential locations in which CPLEX is likely to fail.

- A novel approach has been proposed for dimensionality reduction in MOO problems that is based on an MILP formulation introduced in a previous work by Guillén-Gosálbez (2011b). The method presented relies on decomposing the original MILP into a set of sub MILPs whose solution is used to construct cutting planes for solving the original problem. Numerical results show that our method works efficiently, outperforming in complex problems with a large number of objectives and/or solutions the full space MILP and the exhaustive and greedy algorithms for dimensionality reduction introduced by Brockhoff and Zitzler (2006a). Our tool aims to ameliorate the numerical difficulties arising in the solution of *MOO* problems with a large number of objectives.

- Two improvements have been proposed to be implemented in ϵ-constraint method: To incorporate the objective reduction technique and the pseudo/quasi-random sampling of epsilon parameter. In order to test the capabilities of our approach we have carried out a comparison applying our algorithm to the multi-objective optimization of supply chains considering economic along with environmental concerns. This comparison took into account the number of unique feasible solutions and the hypervolume indicator. Numerical experiments reveal that the best sampling technique is the quasi-random sequence (Halton or Sobol). In addition, the use of a objective reduction method improves the performance of the algorithm from the viewpoints of quantity and quality of Pareto solutions.

5.2 Future Research

- Future work will concentrate on further exploiting the hybridization of LNS and CPLEX, and particularly on investigating how to incorporate the information obtained after solving the MILP sub-problems into the original model in order to expedite the solution of the full space formulation.

- Another key aspect that should be investigated in more detail is the integration of LNS within MOO algorithms in order to enhance their numerical performance.

- As third point, we need to further explore how to systematically select Pareto alternatives according to the decision maker preferences once the Pareto front has been created. This approach could be eventually incorporated into the MOO algorithm itself in order to expedite its solution.

Part III

Last Part: Publications

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

Chapter 6

Large neighbourhood search applied to the efficient solution of spatially explicit strategic supply chain management problems (PUBLISHED)

6.1 abstract

Supply chain management (SCM) has recently gained wider interest in both academia and industry given its potential to improve the benefits of a company through an integrated coordination of all its entities. Optimization problems in SCM are commonly cast as large scale mixed-integer linear programs (MILP) that are hard to solve in short CPU times. This limitation is critical in spatially explicit SCM models since they require a large number of discrete variables to represent the geographical configuration of the network, which leads to complex MILPs. We present herein a novel solution method for this type of problems that combines the strengths of standard branch and cut techniques with the efficiency of large neighbourhood search (LNS). We illustrate the capabilities of this novel approach through its ap-

plication to two case studies arising in energy applications: the design of supply chains (SCs) for bioethanol production and the strategic planning of hydrogen infrastructures for vehicle use.

6.2 Introduction

Supply chain management (SCM) has gained wider interest in both, academia and industry, given its potential to increase the benefits through an efficient coordination of the operations of supply, manufacturing and distribution carried out in a network (Puigjaner and Guillén-Gosálbez, 2008; Naraharisetti et al., 2009). SCM problems can be classified into strategic, tactical and operational according to the temporal and spatial scales considered in the analysis (Fox et al., 2000). In this work we will focus on the strategic level, which deals with decisions that have a long lasting effect on the firm, such as those related with the establishment of new facilities and transportation links between the SC entities.

Traditional SCM models focused on optimizing the economic performance in the private sector. Recently, novel of SCM formulations have emerged that provide decision-making support for public policy makers. Particularly, spatially explicit models have gained wider interest in SCM during the last years. These formulations are particularly suited for strategic SCM problems in which the SC performance shows a strong geographical dependence. This modelling framework maps all possible network configurations within the area of study to a set of geographical locations in which the SC entities (i.e., production and storage facilities) can be established.

This gives rise to large scale MILP models with three types of variables: (1) integers representing the number of facilities opened in a given location; (2) binary variables denoting the existence of transportation links between two sub-regions; and (3) continuous variables that quantify the materials flows and inventory levels. Areas of application of spatially explicit models include the strategic planning of hydrogen SCs for vehicle use (Almansoori and Shah, 2006; Guillén-Gosálbez et al., 2010; Sabio et al., 2009; LI et al., 2008; Kim et al., 2008), and the strategic planning of ethanol SCs (Kostin et al., 2010; Dal-Mas et al., 2011; Giarola et al., 2011; Dunnett et al., 2008; Guillén-Gosálbez and Grossmann, 2009; Duque et al., 2010; Pinto-Varela T. Barbosa-Póvoa and Novais, 2011; Frota Neto et al., 2008; Bojarski et al., 2009; Giarola et al., 2012a).

In spatially explicit SCM models a trade-off exists between modelling accuracy and computational burden. Realistic models require the definition of a large number of discrete variables, which gives rise to large scale MILPs that lead to prohibitive computational times. Decomposition strategies aim to overcome this computational limitation by exploiting the mathematical structure of the problem. This work is to introduces a novel hybrid strategy for the efficient solution of spatially explicit SCM models that combines the complementary strengths of deterministic branch and cut techniques and LNS. We illustrate the capabilities of our method through two case studies that focus on energy applications: the design of hydrogen SCs for vehicle use and the strategic planning of SCs for bioethanol production. Our method, as shown by means of numerical examples, produces near optimal solutions in a fraction of the computational time required by stand-alone determin-

istic branch and cut techniques applied to the original full space MILP. Our solution approach can be easily extended to tackle similar engineering problems with large numbers of discrete decisions, expediting current solution approaches for a certain class of process systems engineering models.

The remainder of this article is organized as follows. Section 6.3 reviews the main solution approaches for SCM problems, with emphasis on mathematical programming techniques and metaheuristics. In section 6.4, we formally define the problem of interest, whereas in section 6.5, we present a generic formulation to solve it. Section 6.6 describes a hybrid approach for the efficient solution of these models that combines branch and cut with LNS. The capabilities of our solution strategy are illustrated in section 6.7 through two case studies based on applications found in the energy sector. The conclusions of the work are finally drawn in section 6.8.

6.3 Literature review: solution methods for SCM

6.3.1 Mathematical programming in SCM

Mathematical programming is probably the prevalent approach for solving SCM problems. A general review on the application of these techniques in SCM can be found in the work by Mula et al. (2010) whereas more specific reviews devoted to process industries have been presented by Grossmann (2005) and Papageorgiou (2009). Particularly, MILP is nowadays the most widely used modelling tool for solving strategic SCM problems. MILP formulations for SCM typically adopt fairly simple linear aggregated repre-

sentations of capacity. Besides avoiding the numerical difficulties associated with dealing with nonlinearities, this simplification permits an easy adaptation to a wide range of industrial applications. In these MILPs, continuous variables represent materials flows and purchases and sales of products, whereas binary variables model tactical and/or strategic decisions related to the network configuration, such as selection of technologies, and establishment of facilities and transportation links (Guillén et al., 2006).

Recently, several spatially explicit SCM models based on MILP have appeared in the literature. Akgul et al. (2011) presented a spatially explicit MILP for the optimal design of a biothanol SC with the objective of minimizing the total cost. The model seeks to optimize the locations of the bioethanol production plants, the biomass and bioethanol flows between regions, and the number of transport units required for transfering these products between several regions of Northern Italy. Dal-Mas et al. (2011) developed a dynamic spatially explicit MILP modeling framework devised to optimize the design and planning of biomass-based fuel SCs under uncertainty in market conditions considering different financial criteria. Mele et al. (2011) presented an MILP to optimize the design of SCs for the combined production of sugar and ethanol. In this work, the problem is formulated as a multi-objective MILP that optimizes simultaneously the economic performance of the network and several LCA metrics (Giarola et al., 2012a,b; Elghali et al., 2007).

Several strategies have been explored for the efficient solution of the aforementioned MILPs arising in SCM. These strategies can be roughly

classified into two major groups: deterministic and stochastic approaches. The former ones provide a rigorous bound on the global optimal solution whereas the latter do not. This is typically accomplished at the expense of larger CPU times, which are required to ensure the quality of the solution found within the desired tolerance.

In addition, deterministic methods can rely on either solving the full space MILP using branch and cut techniques, or decomposing it into sub-problems of smaller size between which an algorithm iterates until a termination criterion is satisfied. We should clarify that in many cases this second type of approaches make also use of branch and cut techniques, but only for solving the sub-problems resulting from decomposing the original MILP and not for the solution of the original MILP itself.

Several decomposition strategies have been devised that exploit the underlying mathematical structure of MILPs arising in SCM. Bok et al. (2000) developed a bi-level decomposition algorithm for an MILP model that maximized the profit of a production-distribution network. This algorithm could halve the solution time compared to the rigorous branch and cut algorithm implemented in CPLEX. Guillén-Gosálbez et al. (2010) introduced a bi-level algorithm for solving the strategic planning of hydrogen SCs for vehicle use. This decomposition method achieved reductions of up to one order of magnitude in CPU time compared with the full space method (the whole model without decomposition, relaxation or approximations) while still providing near optimal solutions (i.e., with less than 1% of optimality gap).

Lagrangean decomposition has also been used in the context of strategic SCM problems. Gupta and Maranas (1999) applied Lagrangean decomposition to solve a planning problem that considered different products and manufacturing sites. The authors reported a solution with an optimality gap of 1.6%, reducing in one order of magnitude the CPU time required by CPLEX 4.0 to find a solution with a gap of 3.2%. You and Grossmann (2010) introduced a spatial decomposition algorithm based on the integration of Lagrangean relaxation and piecewise linear approximation to reduce the computational expense of solving multi-echelon SC design problems under uncertain customer demands. Chen and Pinto (2008) investigated the application of various Lagrangean-based techniques to planning problems that allowed them to reduce considerably the computational burden method while still achieving optimality gaps of less than 2%.

Other solution methods applied to SCM problems have been Benderś decomposition (Geoffrion and Graves, 1974) and "rolling horizon" algorithms based on the original work by Jain and Palekar (2005). The former approach has been applied to strategic/tactical SCM problems with medium-large time horizons (Dogan and Goetschalckx, 1999; Mirhassani et al., 2000; Paquet et al., 2004; Soner Kara and Onut, 2010; Üster et al., 2007). In contrast, rolling horizon algorithms have been primarily used for solving operational SCM problems (Dimitriadis et al., 1997; Elkamel and Mohindra, 1999; Balasubramanian and Grossmann, 2004). In the recent past this approach has also been adapted to deal with strategic SCM problems (Kostin et al., 2011b).

6.3.2 Metaheuristics and hybrid metaheuristics in SCM

Within the second group of mathematical approaches for SCM, we find different types of metaheuristics. Sadjadi et al. (2009) proposed a method to solve the vehicle routing problem (VRP) that combines genetic algorithms with mathematical programming. Chiang et al. (2009) examine the "open" vehicle routing problem with time windows (OVRPTW) using a combination of simulation techniques with tabu search. Amodeo et al. (2009) compared several evolutionary algorithms as applied to SCM models. Delavar et al. (2010) employed genetic algorithms for solving the coordinated scheduling of production and air transportation. Warren Liao and Chang (2010) introduced a method for optimizing inventory levels according to future demand forecasts combining Ant Colony Optimization (*ACO*) with an exponential shooting technique. Baykasoglu and Gocken (2010) proposed a direct solution strategy based on ranking methods of fuzzy numbers and tabu search to solve fuzzy multi-objective aggregate production planning problems.

Some of these metaheuristics have been implemented in several commercial packages and in general-purpose optimization software (Fu, 2002). For instance, the metaheuristic called relaxation induced local search (RINS) (Danna et al., 2005) is currently available in the latest versions of the commercial MIP solvers: LINDO/LINGO (inc. Lindo Systems, 2012) and CPLEX (IBM ILOG, 2012).

In recent years, a new type of algorithms has emerged that combines deterministic and stochastic methods. The idea of integrating different metaheuristic strategies and algorithms dates back to the 1980s. Today, we can observe a generalized common agreement on the advantage of combining components from different search techniques. As a result, there is an increasing tendency of designing hybrid techniques in the fields of operations research and artificial intelligence. The consolidated interest around hybrid metaheuristics is also demonstrated by publications on classifications, taxonomies and overviews on the subject (Blum C. Puchinger et al., 2011).

In general, a hybrid metaheuristic is obtained by combining a metaheuristic with algorithmic components originating from other techniques for optimization (Blum et al., 2008). We may distinguish between two categories. The first relies on designing a solver including components from a metaheuristic into another one, a prominent example being called *memetic algorithms* (Moscato, 1999). The second combines metaheuristics with other techniques that pertains to fields such as operations research and artificial intelligence (van Hoeve and Hooker, 2009; Lodi, 2010). A prominent representative of the first category is the introduction of trajectory methods into population based techniques or the use of a specific local search method into a more general trajectory method such as iterative local search (ILS) (Stützle, 1999). The second category includes hybrid approaches resulting from the combination of metaheuristics with constraint programming (CP) (Shaw, 1998; Applegate and Cook, 1991), integer programming (IP) (Mitrovic-Minic and Punnen, 2009; Prandtstetter and Raidl, 2008), tree-

based search methods and data mining techniques. Both categories contain numerous instances and an exhaustive description is out of the scope of this paper. Particularly, as will be discussed later in this article, in this work we will explore the use of a special type of hybrid method based on the combined used of LNS and branch and cut MIP solvers for tackling strategic SCM problems.

6.4 Problem statement

As mentioned before, we address the solution of MILPs resulting from the formulation of spatially explicit models used in SCM. The problem under study can be formally stated as follows (see Figure C.1). Given are a set of available production, storage and transportation technologies that can be adopted in different locations of a region in order to fulfill the demand of a product of interest. We are also given economic data associated with the establishment and operation of these facilities. The goal of the analysis is to determine the optimal SC configuration, including the type of technologies selected, the capacity expansions over time, and their optimal location, along with the associated planning decisions that optimize a predefined objective function.

6.5 Mathematical Formulation

The strategic planning problem presented above can be posed in mathematical terms as an MILP of the following form:

$$
\begin{aligned}
&\min_{x,Y,N} f(x,Y,N) \\
&\text{s.t.} \\
\text{(M)}\quad &h(x,Y,N) = 0 \\
&g(x,Y,N) \leq 0 \\
&x \subset \mathbb{R}, \quad Y \subset \{0,1\}, \quad N \subset \mathbb{Z}^{+}
\end{aligned}
$$

This generic formulation includes three types of variables: continuous variables x, denoting capacity expansions, production rates, inventory levels and materials flows; discrete variables N, representing the number of transportation units and production and storage facilities opened in a given region; and binary variables Y employed for modelling the establishment of transportation links between two potential locations within the overall region of interest. The inequality and equality constraints, denoted by $g(x,Y,N)$ and $h(x,Y,N)$ respectively, represent mass balances, capacity limitations and objective function calculations. In this work, without loss of generality, we address the solution of two spatially explicit SCM models introduced by the authors in previous works (Mele et al., 2011; Kostin et al., 2011b; Guillén-Gosálbez et al., 2010; Sabio et al., 2010). These multi-period models provide the optimal SC structure along with the capacity expansions over time required to follow a given demand pattern.

For the sake of brevity, a detailed description of the model equations for the two applications addressed in this work (i.e., ethanol and hydrogen

SCs) can be found in Appendix A. Further details on the complete MILP formulations can be found in the original works.

6.6 Solution approach

The difficulty in solving model (M) is highly dependent on the number of integer and binary variables since they are responsible for the combinatorial complexity of the problem. The number of discrete variables required increases with the number of time periods and sub-regions considered in the model. The MILP can be solved via standard branch-and-cut techniques implemented in software packages such as CPLEX. Models accounting for a large number of time periods and/or sub-regions may lead to branch-and-bound trees with a prohibitive number of nodes, thus making the MILP computationally intractable. We next present a hybrid method that combines LNS with standard branch and cut for the efficient solution of (M). Large neighbourhood search was first introduced by Shaw (1998). In LNS an initial solution is gradually improved by alternately destroying and repairing it. This hybrid metaheuristic belongs to a class of algorithms known as Very Large Scale Neighbourhood Search (VLSN) (Ahuja et al., 2002). This approach combines components from different search techniques, and has many potential applications in the fields of operations research and artificial intelligence. Classifications, taxonomies and overviews on the subject can be found in the work by Blum C. Puchinger et al. (2011); Talbi (2002).

All LSN algorithms are based on the observation that searching a large

neighbourhood results in finding local optima of high quality. Specifically, LNS decomposes the original problem into a number of smaller sub-problems that are solved in a sequential way. Each sub-problem emerges from a partial solution, in which some decision variables are fixed and others released. A partial solution defines a neighbourhood of solutions that can be explored rather fast by either tailored (e.g., another heuristic or meta-heuristic) or general purpose algorithms (e.g., branch and cut MIP solvers). A neighbourhood is a function $N : F \rightarrow 2^F$ that assigns to each $a \in F$ a set of neighbouring solutions $N(a) \subseteq F$, where F is the space of all feasible solutions. Hereby, each solution $a' \in N(a)$ is obtained by applying an operator to a, which performs a rather small change to a and hereby creates a new solution a'. Operators or movements are applied in a particular order.

LNS is a general framework that must be adapted to the particularities of the problem under study. Hence, the definition of the large neighbourhood is highly dependent on the problem of interest. In the simplest case, an appropriate portion of the decision variables is fixed to the values that they have in the current solution, and only the remaining "free" variables are considered by the optimization algorithm (typically, a MIP-solver). If the MIP-solver finds an improved solution, it becomes the new current solution, a new large neighbourhood is defined around it, and the process is repeated in subsequent iterations. Obviously, the selection of the variables that remain fixed and the ones that are subject to optimization, respectively, plays a crucial role in the performance of the algorithm. Particularly, the number of free variables directly defines the size of the neighbour-

hood. Too restricted neighbourhoods—that is, sub-problems—are unlikely to yield improved solutions, while too large neighbourhoods might result in excessive running times for solving the sub-problems by the MIP-solver. Therefore, a strategy for dynamically adapting the number of free variables is sometimes used. Furthermore, the variables to be optimized might be selected either purely at random or in a more sophisticated guided way by considering the variables with largest potential impact on the objective function and their relatedness. The section that follows describes the main features of our algorithm.

6.6.1 Algorithm

In this section we will describe the LNS implementation to our particular problem. The algorithm requires the following input data:

- A maximum execution time of the algorithm (t_{max})
- A maximum number of iterations (it_{max})
- A maximum number of variables to be released (n_{max}).
- A maximum number of attempts (m_{max}).

The algorithm works as follows (see Algorithm 17). We generate first the initial solution and calculate the corresponding objective function value. The initial solution is a feasible solution with all the variables fixed. In the main loop, while *end* not equal to true, for each trial m, we do the following. First, we randomly choose a set of n variables V to release. Second,

we copy the solution s to s' and release the n variables from V. Third, we invoke the solver. The solver seeks to improve the solution changing the value of the variables released. The solve invocation gets the run time t employed by the solver, the new value of the objective function fo and a new solution s''. If the objective function value is better than the current value, then we update the best solution, the objective function, and assign to variable *improved* the value of true. Finally, we increase the number of iterations it, and the current time ct, until at least one of them exceeds the predefined limit. When this happens, then the variable *end* is assigned a value of true and the algorithm terminates. The detailed procedures used in our algorithm are described later in the case study section.

6.7 Case studies

Two SCM problems related to energy applications that have attracted an increasing interest in recent years are taken as a benchmark to test the capabilities of our algorithm. For each case study, we first provide a general definition of the problem, and then discus some implementation details of the algorithm before presenting the numerical results.

6.7.1 Design of hydrogen SCs for vehicle use

The first SC design problem has as objective to determine the configuration of a three-echelon hydrogen network for vehicle use (production-storage-market) with the goal of minimizing the total cost. The structure of the

Algorithm 17 LNS for Supply Chain

Require: mdl is the model **AND** $t_{max} > 0$ **AND** $it_{max} > 0$ **AND** $m_{max} > 0$ **AND** $n_{max} > 0$
Ensure: s

1: $\langle s, fo \rangle :=$ initial_solution(mdl);
2: $end :=$ **FALSE** ; $it := 0$; $ct := 0$
3: **while NOT** end **do**
4: $n := 1$; $improved :=$ **FALSE**;
5: **while** $n \leq n_{max}$ **AND NOT** $improved$ **do**
6: $m := 1$;
7: **while** $m \leq m_{max}$ **AND NOT** $improved$ **do**
8: $V := choose\_random\_vars\_to\_release(n)$
9: $s' := release\_vars(s, V)$;
10: $\langle t, fo'', s'' \rangle := solve(mdl, s')$;
11: **if** $better(fo, fo'')$ **then**
12: $fo := fo''$
13: $s := s''$
14: $improved :=$ **TRUE**
15: **end if**
16: $ct := ct + t$
17: $it := it + 1$
18: **if** $ct \geq t_{max}$ **OR** $it \geq it_{max}$ **then**
19: $end :=$ **TRUE**
20: **end if**
21: $n := n + 1$;
22: **end while**
23: $m := m + 1$;
24: **end while**
25: **end while**

three-echelon SC includes a set of plants, where hydrogen can be produced, and a set of storage facilities, where hydrogen is stored before being delivered to final customers. The complete mathematical formulation for this problem can be found in Sabio et al. (2010) and is summarized in Appendix A. From now on, we will refer to this model as HYDROGEN.

The model includes three main types of discrete variables that are relevant in the implementation of our algorithm (see Appendix A):

- **Integer variables** N^{PL}_{igpt}: Number of facilities producing hydrogen in form i using technology p established in location g at period t.
- **Integer variables** N^{ST}_{gst}: Number of storage facilities of type s opened in location g at period t.
- **Binary variables** $X_{gg'lt}$: Equal 1 if there is a link between g and g' using transportation mode l in period t and 0 otherwise.

The initial solution in this case is generated by solving the hydrogen model (MH) with variables NP^{PL}_{igpt}, N^{ST}_{gst} and $X_{gg'lt}$ fixed to the values obtained from a reduced-space model that considers a single time period with a demand equal to the average demands over all the time periods (see Algorithm 18).

6.7.2 Numerical results

In the following subsection we present numerical results that illustrate the performance of our solution method compared with the commercial full

Algorithm 18 Initial solution (MH)

for all g **do**
$D_g := \frac{\sum_t^T D_{gt}}{T}$
end for
Solve MH considering one period ($t = 1$) with demand D_g
Solve MH for all the time periods fixing $\langle N^{PL}_{igpt}, N^{ST}_{gst}, X_{gg'lt}\rangle := \langle N^{PL}_{igp1}, N^{ST}_{gs1}, X_{gg'l1}\rangle$

space branch and cut code implemented in CPLEX. We have selected different instances of the model comprising 2, 4, 6, 8, 10, 12, 14 and 16 time periods, respectively. The MILP size is shown in Table C.1. In Table C.1, we display number of continuous variables, number of binary variables, number of integer variables, number of decision variables (Bin+Int), total number of variables, and finally number of equations. The numerical experiments were performed on a PC Intel (R) Core (TM) Quad CPU Q9550@2.83 GHz and 3 GB of RAM.

First, we tune the algorithm, solving the problems for different values of n and m. In this context, remember that n is the maximum number of variables released and m the maximum number of attempts. Figure C.2 show the results obtained for 10 runs of the algorithm. This figure is a boxplot, a convenient way of graphically depicting groups of numerical data through their five-number summaries: the smallest observation (sample minimum), lower quartile (Q1), median (Q2), upper quartile (Q3), and largest observation (sample maximum). A boxplot may also indicate which observations, if any, might be considered as outliers. That is, in red we show the median or Q2 quartile of the runs, whereas the blue boxes denote the Q1 and Q3

quartiles. The final values selected are highlighted in red color.

Once the algorithm is tuned, we compare its performance with the commercial branch and cut code implemented in CPLEX. Figure C.3 shows for periods $t = 6$ and $t = 12$, the evolution of the lower and upper bounds found by CPLEX as a function of time, along with the values provided by the proposed LNS algorithm as iterations proceed. The time frame considered in the analysis is that for which no further improvement is observed in the LNS. As can be seen, for low time periods (less than or equal to 6), CPLEX outperforms the proposed algorithm, finding better solutions in shorter CPU times. For more than 6 time periods, CPLEX cannot find any solution, whereas the LNS is always able to provide at least one solution. Note that the variability of the results obtained with our algorithm is rather low.

In Table C.2, we provide for each instance being solved, the time at which the best solution calculated by the LNS has been found, the best objective function value in all the runs, the average objective function value and CPU time, and the standard deviation of the objective function and CPU time. Note that in periods 14 and 16 the standard deviation of the CPU time is zero because the algorithm convergence is very fast.

Finally, in Table C.3 displays the optimality gaps of the following solutions: the best solution calculated by CPLEX after 12 hours of CPU time and after the same CPU time provided to the LNS, the best solution found by the LNS and the average solution calculated by the LNS. The optimality

gap is determined from the best solution calculated by CPLEX in 12 hours. Note that in some instances, CPLEX is unable to provide any bound even after the aforementioned CPU time.

6.7.3 Ethanol supply chain model

This second example addresses the design of supply chains for ethanol production. We consider a generic three-echelon SC (production-storage-market). The model is described in detail in Kostin et al. (2011b) and it is summarized in Appendix A. We refer to this model as ETHANOL. The model size is shown in Table C.4, which is equivalent to Table C.1.

The most relevant variables manipulated by our algorithm are the following:

- **Integer variables** NP_{pgt}: Number of factories of technology p established in place g at period t.
- **Integer variables** NS_{sgt}: Number of storage facilities s opened in location g at period t.
- **Binary variables** $X_{lgg't}$: Equals 1 if there is a link between g and g' using transportation mode l at period t and 0 otherwise.

The initial solution for model ETHANOL is generated as follows. We first solve a relaxed MILP in which some integer variables are treated as continuos ones. The solution provided by this relaxed model is then rounded to the nearest integer, and an LP with fixed binaries is finally solved (see

Algorithm 19).

Algorithm 19 Initial solution (ME)

Solve ME relaxing integer variables.
Fix variables to the nearest integer $\langle NT_{lt}, NS_{sgt}, NP_{pgt}, X_{lgg't}\rangle := \langle [NT_{lt}], [NS_{sgt}], [NP_{pgt}], [X_{lgg't}]\rangle$
Solve ME

The results of the tunning are shown in Figure C.4, where the best setting is highlighted in red colour. For low time periods (less than 8), CPLEX outperforms the proposed algorithm. Figure C.5 shows that for 8 time periods our algorithm outperforms CPLEX. For more than 8 time periods, none of the methods performs better than the other one for the whole range of CPU times. Particularly, LNS tend to behave better for short CPU times, whereas CPLEX provides better solutions when long CPU times are considered. Finally, Table C.5 is equivalent to Table C.2, whereas Table C.6 is equivalent to Table C.3.

As observed, LNS provides better solutions (i.e., with better optimality gaps) for problems with large size containing 12, 14 and 16 time periods.

6.8 Conclusions

This work has introduced an efficient hybrid algorithm for spatially explicit SCM models. Our algorithm combines LNS with standard branch-and-cut techniques implemented in the commercial MIP solver CPLEX. The capabilities of the proposed method were illustrated through its application

to two cases studies that focus on energy applications: the design of SCs for sugar and ethanol production and the strategic planning of infrastructures for hydrogen production. Our algorithm was shown to provide near optimal solutions in a fraction of the time spent by the stand-alone branch and cut method implemented in CPLEX and applied to the full space MILP. Furthermore, this method allows identifying feasible solutions even in those cases in which CPLEX fails to converge. Numerical examples demonstrate that our method is particularly suited for tackling large scale problems with high number of time periods and potential locations (and therefore high number of integer and binary variables) in which CPLEX is likely to fail. Future work will concentrate on further exploiting the hybridization of both methods, and particularly on investigating how to incorporate the information obtained after solving the MILP sub-problems into the original model in order to expedite the solution of the full space formulation.

6.9 Acknowledgements

Pedro J. Copado wishes to acknowledge support of this research work from the Spanish Ministry of Education and Science (DPI2008-04099/DPI). The authors wish also to acknowledge support from the CON- 844 ICET (Argentina), the Spanish Ministry of Education and Science (DPI2008-04099/DPI, CTQ2009-14420 and ENE2008-06687-C02-01), and the Spanish Ministry of External Affairs (projects A/8502/07, A/023551/09 and HS2007-0006).

Christian Blum acknowledges support from grant TIN2007-66523 (FOR-

MALISM) and from the *Ramón y Cajal* program of the Spanish Government.

Chapter 7

MILP-based decomposition algorithm for dimensionality reduction in multi-objective optimization: Application to environmental and systems biology problems (PUBLISHED)

7.1 abstract

Multi-objective optimization has recently gained wider interest in different domains of engineering and science. One major limitation of this approach is that its complexity grows rapidly as we increase the number of objectives. This work proposes a computational framework to reduce the dimensionality of multi-objective optimization (MOO) problems that identifies and eliminates in a systematic manner redundant criteria from the mathematical model. The method proposed builds on a mixed-integer linear programming (MILP) formulation introduced in a previous work by the authors. We illustrate the capabilities of our approach by its application

to two SC design problems related to biofuels. Numerical examples show that our method outperforms other existing algorithms for dimensionality reduction.

7.2 Introduction

Multi-objective optimization (MOO) is widely used in many areas of science and engineering for simultaneously optimizing several objective functions subject to some equality and inequality constraints. With the recent advances in software packages and optimization theory, MOO has gained wider interest, being nowadays increasingly used in process systems engineering (PSE). MOO has been applied to several fields like the mean-risk multi-stage capacity investment problem (Claro J., 2012), portfolio selection (Ustun O., 2012), laser cutting of thin sheets of aluminium alloys (Sharma A., 2012), power planning problems in power networks (Alonso M., 2012) and reconfiguration problems in distribution feeders (Niknam T., Fard A. K., 2012), among others.

Unfortunately, the complexity of MOO grows rapidly in size with the number of objectives due to two main reasons (see Deb and Saxena (2005), for further details). First, the solution techniques available for MOO are rather sensitive to the number of objectives. Second, even if we could generate a sufficiently large number of Pareto solutions in an efficient manner, there is still the issue of visualizing and analysing them. To avoid these limitations, most MOO models restrict the optimization task to two or three objectives (López Jaimes et al., 2009). These approaches either omit ob-

jectives, or aggregate them into single metrics defined by attaching weights to them. Both approaches are inadequate, as they change the dominance structure of the problem in a manner such that they might leave solutions that are optimal in the original space out of the analysis.

Dimensionality reduction techniques aim at overcoming these limitations by identifying redundant objectives that can be omitted while still preserving the problem structure to the extent possible. Deb and Saxena (2005) proposed a method based on principal component analysis for reducing the number of objectives in MOO. Brockhoff and Zitzler (2006c) introduced the concept of delta error, an approximation error that arises after removing objectives in a multi-objective optimization problem. These authors formally stated the following two problems: (1) computing a minimum objective subset (MOSS) of a multi-objective problem that does not exceed a certain approximation error (denoted as the δ-MOSS problem); and (2) identifying a minimum objective subset of size k with minimum approximation error (k-MOSS problem). The authors presented both an exact and an approximation algorithm to tackle these problems and applied them to several case studies, showing that substantial dimensionality reductions are possible while still preserving to a large extent the problem structure.

Based on these ideas, Guillén-Gosálbez (2011a) introduced an approach for dimensionality reduction based on an MILP that solves both the k-MOSS and δ-MOSS problems, and which takes advantage of the powerful branch-and-cut algorithms available for MILPs. More recently, Thoai

(2011) proposed an approach to reduce the number of criteria as well as the dimension of a linear MOO problem using the concept of representative and extreme criteria, while López Jaimes et al. (2009) introduced two algorithms for objective reduction in MOO based on a feature selection technique.

In this paper, we present a new decomposition approach for objective reduction that is shown from numerical examples to outperform other existing algorithms Brockhoff and Zitzler (2006c). We apply our algorithm to MOO problems arising in systems biology and sustainable engineering, showing how significant reductions in problem size can be attained without changing too much the problem structure. The article is organized as follows. In section 7.3, we introduce the main concepts and theory behind objective reduction. We then describe the MILP formulation for dimensionality reduction proposed in a previous contribution and discuss its computational limitations (section 7.4). Our computational framework, which is based on this MILP, follows (section 7.5). The capabilities of our approach are tested next (section 7.6), while in the final section (section 7.7) the conclusions of the work are drawn.

7.3 Background

In the next subsections, we present the main concepts behind dimensionality reduction in MOO. We follow the work by Brockhoff and Zitzler (2006a), in which the reader will find further details on this topic.

Let $MO(x)$ be a multi-objective minimization problem of the following form:

$$MO(X) = \min_{x}\{f(x) := (f_1(x), ..., f_k(x)) : x \in X\} \tag{7.1}$$

with k objective functions $f_i := X \longrightarrow \mathbb{R}$, $1 \leq i \leq k$, where each objective function f_i maps a solution $x \in X$ to a value of the function vector $f := (f_1, ..., f_k)$.

We consider the weak Pareto dominance relationship defined as follows : $\preceq_{F'} := \{(x, y) \mid x, y \in X \wedge \forall f_i \in F' : f_i(x) \leq f_i(y)\}$, where F' is a set of objectives with $F' \subseteq F := \{f_1, \ldots, f_k\}$. The following definitions are used in our analysis.

Definition x weakly dominates y ($x \preceq_{F'} y$) with respect to the objective set F' if $(x, y) \in \preceq_{F'}$.

Definition $x^* \in X$ is called Pareto optimal if there is no other $x \in X$ that dominates x^* with respect to the set of all objectives.

In this work we present an algorithm to reduce the dimension (i.e., number of objectives) of $MO(X)$. Given a set of objectives $F := (f_1, \ldots, f_k)$, our goal is to determine a subset F' of F ($F' \subseteq F$) of given cardinality such that the error of removing the objectives not included in F' is minimum. The main ideas underlying dimensionality reduction are illustrated next by means of a simple example.

7.3.1 Illustrative example: objective reduction in environmental problems

Consider the 3 Pareto optimal solutions depicted in Fig. C.6 that minimize 4 different objectives: (f_1, f_2, f_3, f_4). This figure is a parallel coordinates plot (Purshouse and Fleming, 2003) that depicts in the x axis the set of objectives and in the y axis the normalized value attained by each solution. Each line in the figure represents a different Pareto solution. As seen, all of these lines intersect in at least one point, since no solution is dominated by any of the others.

As observed, two objectives can be omitted (i.e.,objective 2 and 3) without changing the dominance structure. This is because $x \preceq_{f_1,f_4} y$ is satisfied if and only if $x \preceq_{f1,f2,f3,f4} y$ is satisfied. Further reductions are not possible without modifying the dominance structure. For instance, if we remove f_1 and f_4, then $x_2 \preceq_{f_2,f_3} x_1$ is satisfied although $x_2 \npreceq_{f_1,f_2,f_3,f_4} x_1$.

As seen, solution x_2 would dominate x_1 in the original 4-dimensional Pareto space $\{f_1, f_2, f_3, f_4\}$ if it showed the same value of f_4 than x_1. The difference between the true value of f_4 in x_2 and that required to dominate x_1 in the original space of objectives can be used as a measure to quantify the change in the dominance structure. For this example, this value, referred to as δ value by Brockhoff and Zitzler, is 0.9. The delta value quantifies the change in the dominance structure of a MOO problem that takes place after removing objectives. Our goal is to determine subsets of objectives of a MOO model that minimize the δ value. This problem was

formally defined in a pioneering work by Brockhoff and Zitzler (2006a). Alternatively, we may also be interested in calculating the minimum number of objectives for a given allowable approximation error. We will refer to these problems as δ-MOSS and k-MOSS problems, respectively. The sections that follow describe in detail our algorithmic approach to tackle them.

7.4 Mathematical Model

A computational framework is proposed for the efficient solution of the k-MOSS and δ-MOSS problems. Our approach builds upon the MILP for dimensionality reduction introduced by Guillén-Gosálbez (2011a). We provide first an overview of this MILP before presenting our methodology for dimensionality reduction (further details on this MILP can be found in the original article).

Given a MOO problem where a set of k objective functions is minimized, we aim at determining an objective subset of given size (i.e., $\mid F' \mid = j$) such that the dominance structure is preserved with a minimal value of δ. For this subset F', the dominance structure of both, the original and reduced sets of objectives, will be the same except for an error equal to δ.

We use the following notation. The parameter $OF(s,i)$ denotes the value of the i objective in solution s. The binary parameter $YP(s',s,i)$ takes the value of 1 if solution s' is better than solution s in objective i (i.e., $OF(s,i) \geq OF(s',i)$) and 0 otherwise. The binary variable $ZO(i)$ is 1

if objective i is removed and 0 otherwise, while the binary variable $ZD(s, s')$ takes the value of 1 if solution s' dominates solution s in the space resulting from removing the objectives i for which $ZO(i) = 1$ (reduced Pareto space) and it is 0 otherwise.

Figure C.7 illustrates an example of this notation. Variable $ZD(s, s')$ is determined by means of the following constraints:

$$\begin{aligned}(k - \sum_i ZO(i)) - k(1 - ZD(s, s')) \leq \sum_i YP(s', s, i)(1 - ZO(i)) \\ \leq (k - \sum_i ZO(i)) + k(1 - ZD(s, s')) \; \forall s \neq s'\end{aligned} \tag{7.2}$$

$$\sum_i YP(s', s, i)(1 - ZO(i)) \leq (k - \sum ZO(i)) - 1 + kZD(s, s') \quad \forall s \neq s' \tag{7.3}$$

Solution s' dominates s in the reduced space, if and only if it is better than s in all of the objectives kept. Therefore, if s' dominates s, then $YP(s', s, i)$ will be equal to 1 for all of the objectives for which $ZO(i) = 0$, and the summation of $YP(s', s, i)$ will equal the number of objectives kept in the reduced space. By adding constraint A.33, we ensure that this will hold if $ZD(s, s')$ is 1. On the other hand, if solution s' does not dominate s, then there will be objectives in which s will be better than s' and others in which the opposite will hold. Consequently, the term $YP(s', s, i)(1 - ZO(i))$ will be necessarily lower than the cardinality of the set of objectives kept, and constraint A.34 will force the binary variable $ZD(s, s')$ to take the

value of 0.

Constraint A.35 specifies the total number of objectives to be omitted:

$$\sum_{i}(ZO(i)) = OB \tag{7.4}$$

Note that the approximation error increases with larger values of OB. The continuous variables $\delta(s, s', i)$ quantifies the error of removing objectives. We define this variable via constraint A.36:

$$(OF(s', i) - OF(s, i))ZOD(i, s, s') = \delta(s, s', i) \quad \forall i, s \neq s' \tag{7.5}$$

In which $ZOD(i, s, s')$ is defined via the following constraints:

$$ZOD(i, s, s') \leq ZO(i) \quad \forall i, s \neq s' \tag{7.6}$$

$$ZOD(i, s, s') \leq ZD(s, s') \quad \forall i, s, s', s \neq s' \tag{7.7}$$

$$ZOD(i, s, s') \geq ZO(i) + ZD(s, s') - 1 \quad \forall i, s \neq s' \tag{7.8}$$

As observed, the value of $\delta(s, s', i)$ is determined only for those solutions s dominated by at least another solution s' in the reduced space of objectives, and only for the omitted objectives i. On the other hand, constraint A.36 forces variable $\delta(s, s', i)$ to take a 0 value when s is Pareto optimal in the reduced space and i is a non-omitted objective. Note that in the latter case variable $ZOD(i, s, s')$ will take a 0 value, making $\delta(s, s', i)$ equal to

zero.

The model seeks to minimize the maximum error of omitting objectives. The overall mathematical formulation can therefore be expressed as follows:

$$\begin{array}{lll} \text{(MOR)} & \min & \max\limits_{s,s',i} \delta \\ & \text{s.t.} & \text{constraints } 2,3 \text{ and } 4 \text{ to } 8 \end{array} \tag{7.9}$$

We can slightly modify model (MOR) in order to calculate the smallest possible set of objectives that preserves the original dominance structure except for an error of δ. This is accomplished by replacing constraint A.35 by Eq. A.41, which imposes an upper bound on the maximum allowable error.

$$\delta(s, s', i) \leq \delta \tag{7.10}$$

The modified model (MOR2) can then be expressed as follows:

$$\begin{array}{lll} \text{(MOR2)} & \max & \sum\limits_{i}(ZO(i)) \\ & \text{s.t.} & \text{constraints 2,3,5 to 8, and 10} \end{array} \tag{7.11}$$

The complexity of these MILPs grows rapidly with the number of solutions and objectives. We present in the section that follows a decomposition strategy to expedite their solution.

7.5 Proposed methodology

The pivotal idea of our method is to decompose the original full space MILP for dimensionality reduction into two sub-problems: a lower bounding and upper bounding sub-problem, between which our algorithm iterates until a termination criterion is satisfied. Furthermore, the solution of these sub-problems, which can be both solved faster than the full-space MILP, are used to construct a set of cutting planes that are used in order to tighten the relaxation of the upper bounding sub-problem, thereby expediting its solution. A brief outline of the algorithm is given below:

7.5.1 Solution Strategy

The proposed decomposition algorithm (see Figures C.8, C.9 and Algorithm 20) solves in each iteration a lower bounding problem (MOR_{μ_p}) and an upper bounding problem until the difference between the lower and upper bounds falls within an epsilon tolerance ϵ.

The lower bounding problem is very similar to the full-space MILP, but as oppose to it, it is defined only for a subset of the Pareto solutions. To this end, we divide the Pareto set of N solutions into subsets containing μ solutions each, with the property that the intersection of these sub-sets contains the original set. Because of the way in which they are constructed, the lower bounding MILPs contain fewer binary variables than the original MILP, and for this reason their combinatorial complexity and computational burden is lower. Furthermore, in each lower level problem we add a set of cutting planes obtained from previous sub-problems of size μ' ,

where $\mu' < \mu$. The lower level problem (MOR_{μ_p}) serves two major purposes. First, it yields a lower bound LB on the global optimum of the full space MILP. Second, it identifies sets of objectives that are likely to yield a good approximation error after being removed from the original MILP. This information is used in the upper bounding problem to expedite its solution.

The upper bounding problem consists of a customized algorithm that calculates the delta error for a given set of objectives removed. This set of objectives is given by the lower bounding problem. The upper bounding problem provides therefore an upper bound UB on the global optimum of the original MILP. Note that calculating the delta error with the customized algorithm is faster than solving the full space MILP. We describe next each of the levels of the algorithm in more detail.

The algorithmic steps of the inner loop are summarized in section 7.5.4. The notation used here is as follows. Variable p denotes the lower bounding problem solved, while μ represents the size of this sub-problem, which is denoted by variable p (hence, MOR_{μ_p} is the problem of size μ solved in iteration p). At the end of each algorithmic iteration, the value of variable μ is kept, while p is increased by one until the value $\frac{N}{\mu}$ is attained, thereby finishing the inner loop. In the outer loop, the value of variable μ is multiplied by r, where $r * \mu$ represents the size of the lower bounding problems in the next iterations. When μ is equal to N, then the outer loop finishes and the algorithm stops giving a solution. At the end of each iteration p, the list of cuts $Cuts_{LB}$ are updated. The list keeps the best cuts derived

from the lower bounding problems. These cuts are sorted according to the delta values attained by the corresponding lower bounding problems. Since the space is limited, we replace the worst cuts be new ones as iterations proceeds.

7.5.2 Lower Level

As previously mentioned, the lower bounding problem corresponds to the original MILP that is defined only for a subset of solutions of size μ. To expedite the solution of these sub-problems, we add cutting planes that are obtained from previous sub-problems already solved. These cutting planes take the following form:

$$a_j \geq b_j \quad \forall j \in CUT_{\mu_p} \tag{7.12}$$

Where CUT_{μ_p} is the set of cuts used in iteration p to solve problems of size μ, and a_j is an auxiliary continuous variable defined as follows:

$$\delta'(s, s') \leq \delta_{\mu_p}, \quad \forall s, s' \in S_{\mu_p} \tag{7.13}$$

$$\delta'(s, s') \leq a_j \quad \forall s, s' \in S_{\mu_p} \quad \forall j \in CUT_{\mu_p} \tag{7.14}$$

$$\delta(s, s', i) \leq \delta'(s, s') \quad \forall s, s' \in S_{\mu_p} \quad \forall i \tag{7.15}$$

The solution of each sub-problem MOR_{μ_p} solved at iteration p provides a lower bound on the global optimum of the full space MILP. This is because

these sub-problems are defined over a reduced number of solutions (i.e., they are relaxations of the full space MILP, and are therefore guaranteed to produce a rigorous lower bound). This property is quite appealing, as it allows approximating the solution of the full space MILP without having to solve it explicitly for problems involving a very large number of solutions and objectives (see proof in Appendix B.1).

7.5.3 Upper Bounding

The upper bounding level determines the delta error for the combination of objectives predicted by the lower bounding MILP. In this level we apply a customized algorithm that calculates the delta error for a given combination of objectives. This algorithm is described in detail in Algorithm 20.

7.5.4 Algorithmic Steps

The detailed steps of the proposed decomposition strategy as follows:

1. Set problem count $p := 1$, upper bound $UB := +\infty$, lower bound $LB := -\infty$, problem size $\mu := \mu_0$, and tolerance to zero ($\epsilon := 0$).

2. Outer loop: while gap condition is satisfied ($UB - LB \geq \epsilon$) and problem size is not exceeded ($\mu \leq N$), otherwise algorithm stops.

   (a) Inner loop: while gap condition is satisfied ($UB - LB \geq \epsilon$) and problem counter is not exceeded ($p \leq \frac{N}{\mu}$), otherwise if gap condition is satisfied ($UB - LB \geq \epsilon$) set $p := 1$ and $\mu := r * \mu$ and Go to step 2.

Algorithm 20 Algorithm to solve the upper level problem, where lines 7 to 12 compute dominance relationship ($dominance[i,j]$) according to fixed variables ZO. Lines 13 to 26 compute delta error δ.

Require: $PS : Matrix[N,K]$ of $\mathbb{R}$, $ZO : Array[i]$ of $Boolean$
Ensure: δ is the maximum delta error of the Pareto set of solutions PS
function CUSTOMIZEDDELTAERROR($PS : Matrix[N,K]$ of $\mathbb{R}, ZO : Array[i]$ of $Boolean$) **return** $\delta : \mathbb{R}$

$\delta, \delta', \delta'' : \mathbb{R}$
$i, j, k : \mathbb{N}$
$dominance : Matrix[N,N] of Boolean$;
$dominance := FALSE$;
$\delta := \delta' := \delta'' := 0$;
for $i := 1; i \leq N; i := i+1$ **do**
  for $j := i+1; j \leq N; j := j+1$ **do**
    $dominance[i,j] := \bigwedge_{ZO[k]} (PS[i,k] \leq PS[j,k])$ and $\overline{\bigwedge_{ZO[k]} (PS[i,k] \geq PS[j,k])}$;
    $dominance[j,i] := \bigwedge_{ZO[k]} (PS[j,k] \leq PS[i,k])$ and $\overline{\bigwedge_{ZO[k]} (PS[j,k] \geq PS[i,k])}$;
  end for
end for
for $i := 1; i \leq N; i := i+1$ **do**
  for $j := i+1; j \leq N; j := j+1$ **do**
    for $k := 1; k \leq K; k := k+1$ **do**
      $\delta' := (PS[j,k] - PS[i,k]) * 1_{dominance[i,j]} * 1_{ZO[k]}$;
      $\delta'' := (PS[i,k] - PS[j,k]) * 1_{dominance[j,i]} * 1_{ZO[k]}$;
      if $\delta \leq \delta'$ **then**
        $\delta := \delta'$;
      end if
      if $\delta \leq \delta''$ **then**
        $\delta := \delta''$;
      end if
    end for
  end for
end for
end function

(b) Solve lower bounding problem MOR_{μ_p} pointed out by p with size μ using cutting planes in list Cut_{LB}. A lower bound on the delta value (δ_{LB}) and a set of objectives to be removed are obtained ($ZO(i)$).

(c) Solve upper bounding problem fixing binary variables ($ZO(i)$) and obtain an upper bound on the delta value(δ_{UB}).

(d) List of cuts Cut_{LB} is updated: A new cut δ_{LB} is then inserted in the corresponding position in the list of cuts Cut_{LB}, replacing a previous cut when the list is filled. The list Cut_{LB} is sorted in descending order of delta value for all of the lower bounding problems solved in previous iterations.

(e) The LB and UB are updated: if $LB < \delta_{LB}$ then $LB := \delta_{LB}$ and if $UB > \delta_{UB}$ then $UB := \delta_{UB}$.

(f) Increase problem counter p by one ($p := p + 1$) and Go to step 2.

Remarks

- As we discussed above, our approach is based on the MILP introduced in a previous work (Guillén-Gosálbez, 2011b), but incorporates some differentiating issues. On the one hand, we decompose the origina MILP via a bi-level algorithm, where the lower level solves sub-problems using the original MILP (that are smaller in size than the original MILP), and the upper level, which employs the results obtained of lower level, is computed by a customized algorithm (see Algorithm 20). On the other hand, our method differs from Guillén-

Gosálbez (2011b) in the inclusion of cuts, which are implemented through equations 7.12 to B.1.

- The lower bounding MILPs can be either solved to global optimality or stopped when an optimality gap is reached. When the second option is selected (which expedites the solution of the sub MILPs), the cutting planes are constructed considering the best possible bound obtained by the branch and cut algorithm (instead of the delta value of the best integer solution).

- Note that we have slightly modified the MILP introduced by Guillén-Gosálbez (2011a) in order to calculate the approximation error in the same manner as proposed by Brockhoff and Zitzler (2006a). Hence, to determine the delta error, we consider any pair of solutions such that one dominates the other in the reduced space regardless of whether both solutions are Pareto optimal in such a reduced domain. Figure C.10 provides an illustrative example on this issue.

- The Pareto solutions must be normalized before the application of the algorithm. There are different alternatives to perform this step. The method followed in this work is described in detail in the Appendix B.2.

7.6 Computational Results

We illustrate the capabilities of our approach using two case studies, which have been solved following the steps summarized in Figure C.11. The first addresses the design of hydrogen supply chains (a problem in the area of

green engineering), while the second deals with the multi-objective optimization of metabolic networks (taken from the field of systems biology). In both cases, our method was compared against the full space MILP model introduced by Guillén-Gosálbez (2011a), and the exact and greedy methods developed by Brockhoff and Zitzler (2006a). All the numerical experiments were conducted on a computer Intel(R) Core (TM) i7-3612QM CPU@ 2.10GHz 2.10GHz and 6GB of memory RAM. We describe next in detail the numerical results obtained in each case.

7.6.1 Design of hydrogen supply chains for vehicle use

This example deals with the optimal design of a hydrogen SC for vehicle use in Spain taking into account economic and environmental concerns. The problem, which was first proposed by Almansoori and Shah (2009), considers different technologies for production, storage and transportation of hydrogen to be established in a set of geographical regions distributed all over the country (see Figure C.12). The goal is to determine the optimal network configuration in terms of its economic and environmental performance. The problem can be formulated as a multi-objective MILP that seeks to minimize the total cost of the network and its environmental impact. In this formulation, integer variables indicate the number of plants and storage facilities to be opened in a specific region (i.e., grid), whereas binary variables are employed to denote the existence of transportation links connecting the SC entities. The environmental impact was calculated through 16 life cycle assessment indicators based on the Eco-indicator 99 methodology. Further details on this case study can be found in Sabio et al. (2010).

Designing efficient hydrogen supply chains requires the simultaneous assessment of several alternatives and identification of the best combination of technologies. These technologies might differ in capital investments, required feedstocks, production cost (Balat and Kirtay, 2010) and environmental performance (Koroneos et al., 2004; Spath and Mann, 2001; Spath, P. Mann, 2001). In this context, optimizing exclusively the economical performance may lead to solutions that do not fully exploit the environmental benefits of moving toward a hydrogen-based energy system De-León Almaraz et al. (2013). It is therefore clear that environmental concerns must be accounted for along with economic criteria in the optimization of hydrogen supply chain. Several works have been presented and we review briefly these. Hugo and Pistikopoulos (2005) developed a model that identifies the optimal infrastructure taking account different hydrogen pathways in Germany in terms of investment and environmental criteria. This model has been extended and considered as a basis for other works (Lin et al., 2008; INGASON et al., 2008; QADRDAN et al., 2008; Guillén-Gosálbez et al., 2010). Murthy Konda et al. (2011) presented a multi-period optimization framework for the design of spatially-explicit and time-evolutionary hydrogen supply networks. Finally, Sabio et al. (2012) addressed the minimization of the total cost along with a set of life cycle assessment (LCA) impacts. The authors performed as well a post-optimal analysis of the results using principal component analysis (PCA) in order to facilitate the interpretation and selection of final alternatives.

We first generated 300 Pareto solutions using the ϵ-constraint method

(Haimes, Y.Y.; Lasdon, L.S.; Wismer, 1971), which solves a set of single-objective problems in which one objective is kept as main objectives while the others are transferred to auxiliary constraints. These solutions were normalized (see Appendix B.2) and then used for dimensionality reduction. Particularly, we solved different instances of the dimensionality reduction problem, in each of which the goal was to eliminate a given number of objectives ranging between 1 to 15. The MILP for dimensionality reduction contains 4485034 variables and 7445102 equations.

Table C.7 shows the numerical results obtained with each of the algorithms (i.e., the full space MILP, the proposed approach, and the two algorithms of Brockhoff and Zitzler (2006a)). Particularly, the table provides the CPU time, the number of iterations (only for the case of our approach) and the optimality gap (only for the case of the greedy algorithm). The MILP and our approach are solved to global optimality (i.e., with an optimality gap of 0%; note that the exhaustive method also provides a solution with a zero gap). On the other hand, the greedy algorithm provides no information regarding the gap of the final solution found. However, for this later algorithm we determine the optimality gap through comparison with the optimal solution calculated with the other exact approaches.

As observed, our method identifies the optimal solution faster than the MILP (Guillén-Gosálbez, 2011a) and than the exhaustive method proposed by Brockhoff and Zitzler (Brockhoff and Zitzler, 2006a) in almost all of the cases solved. Particularly, we get reductions of CPU time in between 1 to 3 orders of magnitude when compared to the other methods that also

guarantee global optimality (i.e., the full space MILP and the exhaustive method). On the other hand, the greedy algorithm is always the fastest approach. Note that despite yielding the global optimum in all of the cases, there is indeed no theoretical guarantee of reaching the global optimum when using this algorithm.

The high performance of the greedy algorithm is due largely to its strategy, which starts with a reduced set of one objective, and then adds to this set the objective that leads to the minimum error, and keeps on doing this until the desired number of objectives kept is reached. This strategy performs very well in practice, but offers no guarantee of convergence to the global optimum.

Note also that the exhaustive method is the best algorithm only for those cases with a small number of objectives kept (i.e., when removing more than 12). Recall that the exhaustive method calculates the error for every possible combination of objectives kept. Hence, when the number of objectives kept is small, and so is the number of potential combinations, the algorithm performs well, since it can determine the delta error of each such combination quite fast. On the contrary, for a large number of potential combinations, we expect the exhaustive algorithm to perform worst than our method.

7.6.2 Multi-objective optimization of metabolic networks

This example deals with metabolic optimization problems arising in systems biology studies (Pozo and Guillén, 2012). Given a metabolic network

(see Figure C.13) described by a GMA model, the goal is to determine a set containing the preferred enzymatic profiles that optimize the synthesis rate of a metabolite at minimum cost (minimum number of changes in these activities, i.e., minimum change in gene expression) and minimum increase in the concentration of intermediate metabolites in the fermentation of Saccharomyces cerevisiae for ethanol production, considering 15 objectives: Objectives $1-8$ correspond to changes in the enzyme expressions that should be minimized so as to make it easier to manipulate genetically the strain of interest and maintain cells homeostasis (enzymes $K1$-$K8$). K1: Hexose transporters, $K2$: Glucokinase/Hexokinase, $K3$: Phosphofructokinase, $K4$: Trehalose 6-phosphate syntase complex (+Glycogen production), $K5$: Glyceraldehyde-3-phosphate dehydrogenase, $K6$: GOL (Glycerol production), $K7$: Pyruvate kynase, $K8$: ATPase; objective 9 is the total cost (the cost of changing the enzyme activities, that should be also minimized); objective 10 is the synthesis rate of ethanol (that should be maximized); and the remaining 5 objectives represent the concentration of metabolites $X1$-$X5$ that should all be minimized and be kept as close as possible to those in the basal state to ensure as well cells homeostasis. $X1$: Internal glucose, $X2$: Glucose-6- phosphate, $X3$: Fructose-1,6-diphosphate, $X4$: Phosphoenolpyruvate, $X5$: Adenosine triphosphate (Pozo and Guillén, 2012).

The importance of multi-objective optimization in metabolic studies has been pointed out by several authors Vera et al. (2003); Liu and Wang (2008); Wu et al. (2011). Pozo and Guillén (2012) presents a strategy of combining multi-objective global optimization and Pareto filters as a manner to identify optimal genetic manipulations in metabolic models. de Hijas-Liste et al.

(2014) apply multi-objective optimization to metabolic pathways. Wang and Wu (2013) introduces a generalized fuzzy multi-objective optimization approach for finding optimal enzyme effects on metabolic network systems. Higuera et al. (2012) apply a multi-objective optimization approach to the allosteric regulation of enzymes using a model of a metabolic substrate-cycle.

For this case, we proceeded similarly as in the previous case. We first generated 300 solutions using the ϵ-constraint method (Haimes, Y.Y.; Lasdon, L.S.; Wismer, 1971). The solutions were next normalized (see Appendix B.2), and we then solved the dimensionality reduction problem, where the goal was to minimize the delta error eliminating a number of objectives ranging between 1 to 14. This case contains 1421528 variables and 3529528 equations.

Table C.8 shows the numerical results obtained with each of the algorithms (i.e., the full space MILP, the proposed approach and the two algorithms of Brockhoff and Zitzler (2006a)). The MILP and our approach are both solved with an optimality gap of 0%. As observed in this case, our method identifies the optimal solution much faster than the MILP (Guillén-Gosálbez, 2011a) and the exhaustive method (Brockhoff and Zitzler, 2006a). Particularly, we get reductions of CPU time of approximately 3 orders of magnitude when compared to the other exhaustive methods that guarantee global optimality. In this case, the greedy algorithm is again always the fastest approach, but recall that it offers no theoretical guarantee of global optimality.

Note the greedy algorithm also shows an exceptional performance in this case, but as already mentioned before, we should emphasize that: (i) it offers no guarantee of convergence to the global optimum; (ii) it provides in this case solutions that are suboptimal.

The exhaustive method is the best algorithm only for the case with 1 objective kept (14 objectives removed). Note that the MILP method fails to provide a solution for those cases with a small number of objectives kept (i.e, more than 11 objectives removed).

7.7 Conclusions

This work has proposed a novel approach for reducing the number of objectives in MOO that is based on an MILP formulation for dimensionality reduction introduced in a previous work by the authors. The method presented relies on decomposing the aforementioned MILP formulation into a set of sub MILPs whose solution is used to construct cutting planes for the original full space MILP. Numerical results show that the method proposed works efficiently, outperforming in complex problems with a large number of objectives and/or solutions the stand alone MILP and the exhaustive and greedy algorithms for dimensionality reduction introduced by Brockhoff and Zitzler (2006a). Our tool aims to ameliorate the numerical difficulties arising in the solution of *MOO* problems with a large number of objectives. Future work will focus on integrating this tool within MOO algorithms in order to enhance their numerical performance.

7.8 Acknowledgements

Pedro J. Copado wishes to acknowledge support of this research work from the Spanish Ministry of Education and Science (DPI2008-04099/DPI). The authors wish also to acknowledge support from the Spanish Ministry of Education and Science (Projects CTQ2012-37039-C02, DPI2012-37154-C02-02 and ENE2011-28269-C03-03).

Chapter 8

Enhancing the ϵ-constraint method through the use of objective reduction and random sequences: application to environmental problems (SUBMITTED)

Summary

The ϵ-constraint method is an algorithm widely used to solve multi-objective optimization (MOO) problems. In this work, we improve this algorithm through its integration with rigorous dimensionality reduction methods and pseudo/quasi-random sequences. Numerical examples show that the enhanced algorithm outperforms the standard ϵ-constraint method in terms of quantity and quality of the Pareto points produced by the algorithm. Our approach, which is particularly suited for environmental problems, which typically contain several redundant objectives, allows dealing with complex MOO models with many objectives.

8.1 Introduction

Multi-objective optimization (MOO) problems arise in all kinds of industrial application areas ranging from production to service industries, entertainment, and many others. The ϵ-constraint method is probably the most widely used approach to solve MOOs. This technique, which was first introduced by Haimes, Y.Y.; Lasdon, L.S.; Wismer (1971), relies on solving a series of single-objective problems in which one objective is kept in the objective function while the others are transferred to auxiliary constraints that bound them within some allowable levels. Each run of these single-objective problems generates a different solution that is guaranteed

to be, at least, weakly Pareto efficient. One of the main advantages of this algorithm is that it can handle non-convex Pareto sets, as oppose to other approaches like the weighted-sum method (Zadeh, 1963) or goal programming (Charnes et al., 1955; Charnes and Cooper, 1961; Ijiri, 1965; Charnes et al., 1967).

Because of its advantageous characteristics, the ϵ-constraint method has found many applications in process systems engineering, system biology, and particularly in problems in which several objectives must be simultaneously optimized. These include the optimal design of supply chains (Yue et al., 2014; Guillén-Gosálbez, 2008), refineries (Gebreslassie et al., 2013; Zhang et al., 2014; Santibañez Aguilar et al., 2014; Murillo-alvarado and Elhalwagi, 2013), reverse osmosis networks (Du et al., 2014), chemical plants (Azapagic and Clift, 1999; Buxton and Pistikopoulos, 2004; Kravanja and Čuček, 2013) and the multi-objective optimization of metabolic networks (Pozo and Guillén, 2012; Banga, 2008).

The main drawback of the ϵ-constraint method is that it is very sensitive to the number of objectives. The standard ϵ-constraint algorithm keeps one objective as main criterion and divides the domain of the rest into equal intervals. A set of single-objective problems is then solved for the limits of those intervals, generating in each run a different Pareto point. If we choose p partitions for each objective, we will have to solve $2+p-1$ problems for the case of two objectives, $3+(p-1)^2$ for three, and so on (in general, $k+(p-1)^{k-1}$, being k the number of objectives being optimized, and p the number of partitions, (note that to determine the limits of the

intervals, we first need to optimize each single objective separately). Hence, the complexity of this method grows exponentially in size with the number of objectives, and can get out of hand very easily for problems with several objectives. For this reason, the overwhelming majority of works in the literature that make use of the ϵ-constraint method optimize two objective functions, and three at most, but very seldom more than three.

In this work we introduce an enhanced ϵ-constraint method that integrates two main ingredients: (i) a rigorous objective reduction technique that eliminates redundant objectives from the search (and which simplifies the generation and analysis of the Pareto solutions); and (ii) pseudo/quasi-random sequences (i.e., uniform distribution and Sobol and Halton sequences), which are employed to generate in a more efficient manner the values of the epsilon parameters used in the single-objective problems solved by the ϵ-constraint algorithm. We illustrate the capabilities of our approach through its application to two supply chain design problems, in which we optimize the economic performance along with a set of environmental metrics quantified following LCA principles. Numerical results show that, compared to the standard ϵ-constraint method, the enhanced algorithm provides solutions of better quality in less computational time. Our approach can find many applications in process systems engineering, but it is particularly suited for environmental problems in which several environmental objectives tend to be redundant.

The article is organized as follows. In section 8.2, we introduce the ϵ-constraint method. We then describe the improvements proposed in this

paper (section 8.3). The capabilities of our approach are tested next (section 8.4), while in the final section (section 8.5) the conclusions of the work are drawn.

8.2 Mathematical background

Let us consider the following *MOO* problem:

$$\begin{aligned} &\min_x F(x) := (f_1(x), ..., f_k(x)) \\ &\text{s.t. } g_j(x) \leq 0, && j = 1, 2, \ldots, m, \\ &\quad h_l(x) = 0, && l = 1, 2, \ldots, e, \end{aligned} \tag{8.1}$$

with k objective functions $f_i := X \longrightarrow \mathbb{R}$, $1 \leq i \leq k$, where each objective function f_i maps a solution $x \in X$ to a value of the function vector $F(x) := (f_1(x), ..., f_k(x))$, m is the number of inequality constraints, and e is the number of equality constraints. $x \in X$ is a vector of design variables (also called decision variables).

Definition x weakly dominates y with respect to the objective set $F'(x \preceq_{F'} y)$ if $(x, y) \in \preceq_{F'}$.

Definition $x^* \in X$ is called Pareto optimal if there is no other $x \in X$ that dominates x^* with respect to the set of all objectives.

The ϵ-constraint method relies on solving a set of single-objective auxiliary problems in which one objective is kept as main objective while the

others are transferred to auxiliary constraints that bound them within some limits. This algorithm solves single-objective problems of the following form:

$$\begin{array}{llll} \min f_i(x) & \text{where} & 1 \leq i \leq k & \\ \vdots & \vdots & \vdots & \vdots \\ \text{s.t. } f_j(x) \leq \epsilon_j^r & \text{where } i \neq j & 1 \leq j \leq k & 1 \leq r \leq p \\ x \in X & \text{where} & LB_j \leq \epsilon_j^r \leq UB_j & \end{array} \tag{8.2}$$

As already mentioned, this method is quite sensitive to the number of objectives. In general, the number of iterations is equal to $k + (p-1)^{k-1}$, being p the number of partitions defined for every objective and k the number of objectives. It is therefore clear that the computational burden grows rapidly (in fact, exponentially) in size with the number of objectives. In the sections that follow we introduce some improvements that expedite the application of this algorithm.

8.3 Proposed approach: enhanced ϵ-constraint method

The ϵ-constraint algorithm is expedited through the integration of two main ingredients: a dimensionality reduction method and the use of pseudo/quasi-random sequences. Each of these techniques is next explained in detail before providing a detailed description of whole algorithm.

8.3.1 Dimensionality reduction

Dimensionality reduction methods eliminate redundant objectives in *MOO* problems (see Figure C.14). Consider the general *MOO* model introduced before. The goal of dimensionality reduction algorithms is to find a subset F' of objectives functions pertaining to the original set F with the following property: when we optimize the problem in the reduced domain of objectives F' (rather than on the original domain F), we will generate (in less CPU time) a Pareto front that is very close to the " true " Pareto front of the original problem.

Different dimensionality reduction strategies have been proposed in the literature. In a seminar work, Deb and Saxena (2005) introduced an algorithm for objective reduction based on principal component analysis. An alternative algorithm to reduce the number of objectives that is based on a feature selection technique was introduced by López Jaimes et al. (2008). Brockhoff and and Zitzler were the first to formally state the problem of reducing the number of objectives in MOO. Their seminar work introduced the concept of error of the approximation obtained after removing objectives in a MOO problem, and formally stated the following two problems: (1) computing a minimum objective subset (MOSS) of a multi-objective problem that does not exceed a certain approximation error (denoted as the δ-MOSS problem); and (2) identifying a minimum objective subset of size k with minimum approximation error (k-MOSS problem). The authors presented also an exact and an approximation algorithm to tackle these problems that were applied to well-known test case studies, show-

ing that substantial dimensionality reductions are possible while keeping the approximation error low (Brockhoff and Zitzler, 2006b). Bearing these ideas in mind, Guillén-Gosálbez (2011a) introduced an approach for dimensionality reduction based on a mixed-integer linear program (MILP) that solves both the k-MOSS and δ-MOSS problems, and takes advantage of the powerful branch-and-cut algorithms available for MILP (Guillén-Gosálbez, 2011a; Copado-Méndez et al., 2014). More recently, Thoai (2011) proposed ways to reduce the number of objectives and dimension of a linear multiple criteria optimization problem using the concept of so-called representative and extreme criteria (Thoai, 2011).

In this paper, we use the dimensionality reduction method introduced by Guillén-Gosálbez (2011a) and later enhanced by Copado-Méndez et al. (2014), to expedite the performance of the ϵ-constraint method. We next provide a very simple example of dimensionality reduction. Further details of this method can be found in the original publication.

Consider the 3 Pareto optimal solutions depicted in Figure C.14 that minimize 4 different objectives: (f_1, f_2, f_3, f_4). This figure is a parallel coordinates plot (Purshouse and Fleming, 2003) that depicts in the x axis the set of objectives and in the y axis the normalized value attained by each solution. Each line in the figure represents a different Pareto solution. As seen, all these lines intersect in at least one point, as no solution is dominated by any of the others. As observed, two objectives can be omitted (i.e., objective 2 and 3) without changing the dominance structure. This is because $x \preceq_{f_1,f_4} y$ is satisfied if and only if $x \preceq_{f1,f2,f3,f4} y$ is satis-

fied. Further reductions are not possible without modifying the dominance structure. For instance, if we remove f_1 and f_4, then $x_1 \preceq_{f_2,f_3} x_2$ is satisfied although $x_1 \not\preceq_{f_1,f_2,f_3,f_4} x_2$. As seen, solution x_2 would dominate x_1 in the original 4-dimensional Pareto space $\{f_1, f_2, f_3, f_4\}$ if it showed the same value of f_4 than x_1. The difference between the true value of f_4 in x_2 and that required to dominate x_1 in the original space of objectives can be used as a measure to quantify the change in the dominance structure. For this example, this value, referred to as δ value by Brockhoff and Zitzler, is 1.5. The delta value quantifies the change in the dominance structure of a MOO problem that takes place after removing objectives. Our goal is to determine subsets of objectives of a MOO model that minimize the δ value. This problem was formally defined in a pioneering work by Brockhoff and Zitzler (2006a). Alternatively, we may also be interested in calculating the minimum number of objectives for a given allowable approximation error. We will refer to these problems as δ-MOSS and k-MOSS problems, respectively. The sections that follow describe in detail our algorithmic approach to tackle them.

8.3.2 Random sequences

Objective-reduction techniques might be unable to eliminate a significant amount of objectives in cases where few redundant criteria exist. If we partition the domain of each objective into equal intervals during the application of the ϵ-constrain method, we might still need to solve a very large number of single-objective models. Moreover, most of them might be either unfeasible or produce the same Pareto solution. To overcome this limitation to the extent possible, we propose to generate the epsilon pa-

rameters using random sequences. Particularly, in this paper we focus on the following strategies: a pseudo-random sequence (uniform distribution), and the Halton and Sobol sequences (quasi-random).

The pseudo-random sequence is well known, and therefore we will not get deeper into its details. The Halton and Sobol sequences (also called low-discrepancy or quasi-random sequences) are useful in numerical integration, as well as in simulation and optimization. Surveys of applications of low-discrepancy sequences can be found in Fox (1986); Bratley and Fox (1988); Bratley et al. (1992).

In the context of our work, the aforementioned sequences are used, to generate the values of the epsilon parameters in a more effective manner. The goal is to spread these values as much as possible so they cover a wider region of the search space, which will eventually lead to a better Pareto front.

The concept of *low-discrepancy*, which is introduced next, will be used in the ensuing sections for justifying the use of the sequences:

Definition The *discrepancy* D_p^k of point set $\{x_i\}_{i=1\ldots p} \in [0,1)^k$ is:

$$\begin{aligned} &D_p^k = \sup_E \left| \tfrac{A(E,p)}{p} - \lambda(E) \right| \\ &\text{where } E = [0,t_1) \times \ldots \times [0,t_k),\ 0 \leq t_j \leq 1,\ j = 1,\ldots k, \\ &\lambda(E) \text{ is the hypervolume of } E, \\ &A(E,p) \text{ is the number of } x_i \text{ contained in } E, \\ &p \text{ is number of points and } k \text{ is the number of dimensions.} \end{aligned} \tag{8.3}$$

In other words, E is a set of hyperectangles, $\lambda(E)$ is the hypervolume, $\frac{A(E,p)}{p}$ is the percentage of points x_i, that fall in E, and discrepancy is

the largest difference between $\frac{A(E,p)}{p}$ and $\lambda(E)$, that is, the discrepancy measures the error in the hypervolume estimation (see Figure C.15).

Definition A sequence of p points is considered a *low-discrepancy* sequence, if its $D^k(p)$ is in $\mathrm{O}\big(log(p)\big)^k$.

We next describe each of the sequences used in our work in detail.

8.3.2.1 Halton sequence

The Halton sequence, which was introduced by Halton (1960), can be used like a random number generator to produce points in the interval $[0, 1]$. The standard approach shown below is employed in our work. More details on this strategy can be found in Faure and Lemieux (2010). In one dimension, the standard Halton sequence is generated by choosing a prime number r ($r \geq 2$), and decomposing an integer g in terms of the base r:

$$\begin{aligned} & g = \sum_{l=0}^{L} b_l(g) r^l \\ & \text{where } 0 \leq b_l(g) \leq r-1 \text{ and } r^L \leq g \leq r^{L+1} \\ & L \text{ is the length of the integer } g \text{ in base } r. \end{aligned} \tag{8.4}$$

The Halton value $\varphi(g)$ is generated by multiplying each digit in base r of g in reverse order by the power of r^{-l-1}:

$$\begin{aligned} & \varphi(g) = \sum_{l=0}^{L} b_l(g) r^{-l-1} \\ & \text{where } 0.b_0(g), \ldots, b_L(g) \text{ are digits of } g \text{ in base } r \text{ in reverse.} \end{aligned} \tag{8.5}$$

For instance, suppose we want to obtain the fourth element ($g = 4$) of the Halton sequence in base 2. The integer 4 is written in base 2 as $4 = 1*2^2+0*2^1+0*2^0$. According to equation 8.5, this number in reverse is 0.001, and then $0 * 2^{-1} + 0 * 2^{-2} + 1 * 2^{-3} = \frac{1}{8}$. Therefore $\frac{1}{8}$ forms the fourth number of the Halton sequence.

The Halton sequence for k dimensions is generated by the method shown above, but taking k first prime numbers as base.

8.3.2.2 Sobol sequence

The other low-discrepancy sequence applied here is the Sobol sequence, which was introduced by Sobol (1967). We provide an informal description below, while theoretical aspects can be found in Sobol (1967) and Bratley and Fox (1988). For simplicity, we assume the case of one dimension in which we aim to generate the g Sobol value $\varphi(g)$, with low discrepancy over the unit interval. To begin, we need to obtain the set of *direction numbers* $v_1, v_2, \ldots$, where each v_i is a binary fraction with this form $v_i = 0.v_{i1}v_{i2}\ldots$. Alternatively, $v_i = \frac{m_i}{2^i}$, where m_i is an odd integer, $0 < m_i < 2^i$. The direction numbers come from a primitive polynomial belonging to the finite field $\mathbb{Z}_2$ [1], which presents the form $P \equiv x^d + a_1 * x_1^{d-1} + \ldots + a_{d-1} * x + 1$ where $a_i \in \{0, 1\}$ and P is a primitive polynomial of degree d in $\mathbb{Z}_2$. The polynomial coefficients are used to define the recurrence for calculating v_i,

[1] The field $\mathbb{Z}_2$ is the finite set {0,1}.

thus:

$$\begin{aligned} &v_i = a_1 * v_{i-1} \oplus a_2 * v_{i-2} \oplus \dots \\ &\oplus a_{d-1} * v_{i-d+1} \oplus v_{i-d} \oplus (\tfrac{v_{i-d}}{2^d}),\ i > d \\ &where \oplus \text{ denotes exclusive-or operator.} \end{aligned} \tag{8.6}$$

Therefore, this recurrence can be rewritten in terms of m_i

$$\begin{aligned} &m_i = 2 * a_1 * m_{i-1} \oplus 2^2 * a_2 * v_{i-2} \oplus \dots \\ &\oplus 2^{d-1} * a_{d-1} * m_{i-d+1} \oplus 2^d * v_{i-d} \oplus v_{i-d},\ i > d \\ &\text{where } \oplus \text{ denotes exclusive-or operator.} \end{aligned} \tag{8.7}$$

The g Sobol value $\varphi(g)$ is obtained as follows:

$$\begin{aligned} &\varphi(g) = \varphi(g-1) \oplus v_c \\ &\text{where } c \text{ is right-most zero bit of Gray code representation of } g \end{aligned} \tag{8.8}$$

The Gray code is a binary numeral system where each number differs from the previous one in only one bit, and it can be computed as follows:

$$g = \dots g_3 g_2 g_1 := \dots b_3 b_2 b_1 \oplus \dots b_4 b_3 b_2 \tag{8.9}$$

To start the recurrence, we take $\varphi^0 = 0$. For example, we take as primitive polynomial

$$x^3 + x + 1$$

then

$$m_i = 2 * m_{i-2} \oplus 8 * m_{i-3} \oplus m_{i-3}$$

and take

$$m_1 = 1, m_2 = 3 \text{ and } m_3 = 7$$

. Therefore

$$m_4 = 12 \oplus 8 \oplus 1 = 1100 \oplus 1000 \oplus 0001 = 0101 = 5$$

,

$$v_1 = \frac{1}{2^1} =_{binary} 0.1$$

$$v_2 = \frac{3}{2^2} =_{binary} 0.11$$

$$v_3 = \frac{7}{2^3} =_{binary} 0.111$$

$$v_4 = \frac{5}{2^4} =_{binary} 0.0101$$

$$\varphi(g(1)) = \varphi(g(0)) \oplus v_1 = 0 \oplus 0.1 = 0.1 =_{decimal} \frac{1}{2} \ (c = 1 \text{ and } g(1) = 0001)$$

$$\varphi(g(2)) = \varphi(g(1)) \oplus v_2 = 0.1 \oplus 0.11 = 0.01 =_{decimal} \frac{1}{4} \ (c = 2 \text{ and } g(2) = 0011)$$

$$\varphi(g(3)) = \varphi(g(2)) \oplus v_1 = 0.01 \oplus 0.1 = 0.11 =_{decimal} \frac{3}{4} \ (c = 1 \text{ and } g(3) = 0010)$$

$$\varphi(g(4)) = \varphi(g(3)) \oplus v_3 = 0.01 \oplus 0.111 = 0.101 =_{decimal} \frac{5}{8} \ (c = 3 \text{ and } g(4) = 0110)$$

And so on ...

This method can be extended to k dimensions by choosing k different primitive polynomials, computing their direction numbers and generating each component $\varphi(g, k)$ separately.

8.3.2.3 Detailed steps of the algorithm

Having presented the ingredients of our algorithm, we next describe in detail how it works. The first step is to generate an initial set of Pareto points on the basis of which we will perform the objective-reduction analysis. To this end, we apply a heuristic approach consisting of solving a set of bi-criteria problems in each of which we trade-off one objective (when dealing with environmental problems, we will choose the economic performance), against each of the remaining criteria (the environmental indicators) separately. For each of these bi-criteria problems, the ϵ-constraint method is run for a given number of iterations, producing a set of Pareto solutions.

The objective reduction method is then applied to identify and eliminate redundant objectives. A set of values of the epsilon parameters are next generated using the pseudo-random sequences. The single-objective problems are finally solved for these parameters values. Hence, the detailed steps of the enhanced ϵ-constraint algorithm (Algorithm 21) are the following:

Algorithm 21 The ϵ-constraint method presented.

1. **Step:** Initialization:
   (a) **Step:** Compute bounds for each objective function $f_j(x)$, optimizing each individual objective separately. Store the best LB_j and worst UB_j values of each objective function obtained in these optimizations.
   (b) **Step:** Generate an initial solutions set $|I|$ using a bi-criteria algorithm. In each run of the bi-objective algorithm, trade-off one objective against another. The values of the auxiliary epsilon parameters are obtained by splitting the auxiliary intervals using a random sequence.
   (c) **Step:** Apply the objective reduction method to the solution set I in order to identify the set of non-redundant (essential) objectives (F'), assuming a maximum allowable approximation error.
2. **Step:** Apply the enhanced ϵ-constraint method taking into account only those objectives contained in F' as follows:
   (a) **Step:** Choose an objective f_i as main objective and transfer the remaining objectives (i.e., $f_i \neq f_j$) to auxiliary constraints, giving rise to the problem expressed by Equation 8.2.
   (b) **Step** Calculate the epsilon parameter values (ϵ_j) for each objective j by splitting the interval $[LB_j, UB_j]$ into $N-1$ sub-intervals according to a pseudo/quasi-random sampling approach
   discussed in previous sections.
   (c) **Step** Solve the resulting single-objective problems (see Equation 8.2) for each set of epsilon parameters a total of $N^{|F'|-1}$ problems must be solved.

8.3.2.4 Remarks

- The outcome of the objective-reduction algorithm depends on the number of Pareto points used in the analysis. Experimental results generated for different problems indicate that a set of Pareto points of small/medium size suffices to identify the redundant objectives. Furthermore, our dimensionality reduction algorithm provides similar results regardless of the region of the search space from where the Pareto points are taken.

- The general approach presented in this article can be easily extended to work with other MOO solution algorithms.

- It is possible to define an outer loop in the calculations in order to repeat the application of the dimensionality reduction algorithm with more Pareto points as iterations proceed and more Pareto points are generated in the reduced domain. In practice, we found that such a loop does not improve significantly the performance of the overall method, since the algorithm for dimensionality reduction provides consistent results for a small number of Pareto points.

8.4 Experiments and results

We test the capabilities of our method through its application to two supply chain design problems, where we optimize the economic performance against a set of life cycle assessment metrics . These problems are formulated as multi-objective MILPs. Details on them can be found in Kostin et al. (2012) and Sabio et al. (2010). The case studies that we deal with

are based on a superstructure of alternatives that accounts for a set of available technologies to produce, store and deliver ethanol/hydrogen (see Figure C.16).

Two multi-objective MILPs were derived, one for each example, and implemented in GAMS, where the objective reduction algorithm was also coded. The MILPs were solved by CPLEX 12.5.1.0 on an Ubuntu 13.10 with an Intel i7-4770, 3.4GHz 8-cores processor, and 16 GB of RAM. The Halton and Sobol sequences were implemented in GAMS as an extrinsic function.

The objective reduction was carried out using the algorithm presented in a previous publication (see Section 8.3.1). To run this algorithm, we used a Pareto set generated by solving a set of 2-objective problems in each of which we applied pseudo-random sequences to generate the values of the epsilon parameters.

Particularly, the following approaches are tested in the case studies (see Algorithm 22):

- Standard ϵ-constraint (equidistant epsilon values, see Algorithm 24).
- Standard ϵ-constraint with pseudo-random epsilon values.
- Standard ϵ-constraint with Halton epsilon values.
- Standard ϵ-constraint with Sobol epsilon values.
- ϵ-constraint integrated with objective reduction and using equidistant epsilon values (see Algorithm 24).
- Our ϵ-constraint integrated with objective reduction and using pseudo-random epsilon values.
- Our ϵ-constraint integrated with objective reduction and using Halton epsilon values.
- Our ϵ-constraint integrated with objective reduction and using Sobol epsilon values.

In order to assess the performance of each algorithm, we used two different indicators that quantify the quality of the Pareto front produced by them. The first is the number of unique Pareto solutions generated in a given time. The second is the quality of the Pareto front, which is quantified using the hypervolume indicator. The detailed calculations performed in this sections are as follows:

Algorithm 22 Procedure comparison

1. **Step:** Run our ϵ-constraint described in Algorithm 21 following the next sampling techniques:
   (a) **Step:** Split epsilon parameters in equidistant sub-intervals (see Algorithm 24).
   (b) **Step:** Split epsilon parameters by means of the pseudo-random method.
   (c) **Step:** Split epsilon parameters by means of the Halton sequence.
   (d) **Step:** Split epsilon parameters by means of the Sobol sequence.
2. **Step:** Run the ϵ-constraint method without eliminating objectives, that is, taking into account all of the objectives (full space) and applying the same sampling techniques described above.
3. **Step:** Carry out a feasibility analysis in order to assess the number of new (unique) feasible solutions per iteration and per time unit. The analysis is depicted in Figures C.17a-C.17b and C.18a-C.18b.
4. **Step:** Compute the hypervolume indicator per time unit (Figures C.17c and C.18c).

The hypervolume indicator (or S-metric, Lebesgue measure), which was introduced by Zitzler and Thiele (1998), is regarded as a fair measure of the quality of a Pareto front, given its convenient mathematical properties (Zitzler et al., 2003). Formally, the hypervolume indicator is defined as follows:

Definition Given a set P of Pareto points ρ, the hypervolume indicator is

the size of the polytope π^d where

$$\pi^d = \{x \in \mathbb{R}^{d+} : \rho \preceq x \text{ for some } \rho \in P\}$$

This polytope corresponds to the space which is dominated by at least one point in the set P. The dominated hypervolume is calculated with respect to a reference point which is chosen to coincide with the nadir point.

The calculation of the hypervolume indicator is a particular case of the Klee's measure problem, which was formulated by Klee (1977). The best algorithm currently known for a d-dimensional space is $\mathrm{O}\big(Nlog(N+N^{\frac{d}{2}})\big)$, and was obtained by Beume (2006). The complexity of computing the hypervolume grows rapidly in size with the number of objectives and Pareto points. For our experiments, the approximation algorithm developed by Everson et al. (2002) and implemented in Matlab was used. This approach uses Monte Carlo sampling to approximate the hypervolume and avoid expensive calculations. A brief outline of this algorithm is given in Algorithm 23.

8.4.1 Sustainable planning of Ethanol supply chain

The first example determines the configuration of a three-echelon bioethanol network and associated planning decisions that maximize the net present value and minimize the environmental impact. Decisions to be made include the number, location, and capacity of the production plants, and warehouses to be set up in each region, their capacity expansion policy

Algorithm 23 Hypervolume algorithm

1. Input: normalized Pareto Set P, α natural
2. Output: hypervolume π^d
3. Generate a pseudo-random set $D \in [0,1]^d$ with the cardinal of α.
4. Count the number of points $x \in D$, which are dominated by any point in $\rho \in P$
5. Compute the fraction of points that are dominated.

for a given forecast of prices and demand over the planning horizon, the transportation links and transportation modes that need to be established in the network, and the production rates and flows of feedstocks, wastes, and final products (Kostin et al., 2012).

The mutli-objective optimization problem contains 6 objectives: the net present value (economic objective and abbreviated as NPV), and the individual categories considered in the Eco-indicator 99, namely, damage to human health (DHH), damage to eco-system quality (DTE), and damage to resources (DTR), along with the global warming potential (GWP), and the Eco-indicator 99 itself (EI99). Note that the later 5 objectives are used to quantify the environmental performance. The MILP contains $36,629$ continuous variables, $10,962$ discrete variables, and $48,588$ equations.

Following our approach (see Algorithms 21 and 22), we first generated 100 Pareto solutions solving bi-criteria problems in the original search space where 20 of these solutions were generated by the 2-objective model NPV-DTE, other 20 with the 2-objective model NPV-DTR, and so on. We

Algorithm 24 Equidistant method

1. Input: it^{MAX},*alphabet* $(\{a, b, c \ldots\})$
2. Output: Pareto set solutions
3. $w \leftarrow$ permutation with repeats randomly chosen from *alphabet*. $(w = \langle \epsilon_2, \ldots, \epsilon_k \rangle)$
4. Loop: while number iteration it is less than a maximum number iterations $(it <= it^{MAX})$.
   (a) Solve ϵ-constraint using w as epsilon parameters.
   (b) $w \leftarrow$ next word in lexicographic order.
   (c) Go to step 2.

then applied the objective reduction (δ-MOSS) algorithm (Copado-Méndez et al., 2014), which took 2 seconds to identify 2 redundant objectives considering an approximation δ equal to 0. We next ran 2,000 iterations of the enhanced ϵ-constraint method and the standard ϵ-constraint method using each sampling technique (equidistant, random, Halton and Sobol).

The equidistant sampling was carried out by optimizing the NPV against the different environmental objectives considering 5 partitions for each objective, so that the 4 interior partition points of each objective j were labelled with alphabet letters $a_j \ldots d_j$, taking into account each sub-problem as a permutation with repeats on the letters $a \ldots d$ in the lexicographic order[2] (see Algorithm 24). As observed in Algorithm 24, firstly, we choose a permutation randomly as seed, and then, we solve the next 2,000 sub-

[2]lexicographic order is the same as dictionary order.

problems considering the environmental objectives in the order presented above.

The results are shown in Figure C.17 and Tables C.9 and C.10. Particularly, in Table C.9 we show the number of unique feasible solutions, the total time required to run $2,000$ iterations and the hypervolume obtained for each approach, whereas in Table C.10 some intermediate results (i.e., at $2,000$ seconds) are shown. Note that the results in both Tables (C.9 and C.10) have been sorted first by the hypervolume indicator and second by the number of unique feasible solutions (both in descending order).

As seen in Tables C.9 and C.10 and Figure C.17, the best approach according to the hypervolume obtained after the 2000 iterations is the ϵ-constraint method with Halton sampling run in reduced space (RS), followed by the Sobol and random executed in the RS, the Sobol and Halton run in the full space (FS), the equidistant sampling in the RS, the random sampling in the FS and the equidistant sampling in the FS. We next proceed to analyze in detail these results:

- **Full space (FS) against reduced space (RS)**: the ϵ-constraint method run in the RS solves single-objectives problems with less auxiliary constraints than the FS approach. This leads to significant CPU time reductions for each iteration, as seen in Figure C.17 and in the column "Total Time" of Table C.9. Besides, having less constrained objectives reduces the chances of having one constraint acting as a bottle neck. This prevents the algorithm from calculating repeatedly the same solution over the $2,000$ iterations. This increases significantly the amount of new unique feasible solutions, having in turn a positive effect on the hypervolume indicator. Note that the gain ob-

tained by reducing objectives in terms of hypervolume is higher than using only pseudo/quasi-sampling sequences (see Table C.9 and C.10).

- **Equidistant sampling against pseudo/quasi-random sampling**: as we observe in Tables C.9 and C.10 and Figure C.17, the equidistant sampling is the slowest even in the RS. The reason why this happens is that solvers firstly try to identify solutions involving active constraints yet when epsilon parameters of the epsilon constraint are chosen equidistantly, it is more unlikely that there is no feasible integer solution satisfying the constraint as an equality (Ehrgott and Ryan, 2003). In practice, the branch and bound method needs to perform a large number of iterations until it can guarantee that the problem is unfeasible. On the other hand, when the epsilon parameters values are chosen by pseudo/quasi-random sequences, it is more likely that there will be an integer solution satisfying the auxiliary epsilon constraint as an equality (Ehrgott and Ryan, 2003). Note that, even in the FS, the number of unique feasible solutions obtained with psuedo/quasi-random sequences is greater than that obtained with the traditional (i.e., equidistant) ϵ-constraint in RS.

- **Pseudo-random against quasi-random sampling**: Halton and Sobol sequences (see Section 8.3.2) choose epsilon parameters values with random properties but more uniformly distributed than pseudo-random sequences. This is because the quasi-random sequences choose the next parameter value taking into account the previous one, and because of this not all the values have the same probability of occur-

rence. On the other hand, the epsilon parameters chosen by pseudo-random sequences are all equiprobable. If we compare the percentage of epsilon parameters values giving rise to Pareto frontier points, we will see that it is higher for quasi-random sampling than for pseudo-random sampling, as we can observe in Tables C.9 and C.10 where quasi-random sampling obtained more feasible solutions.

- **Number of feasible solutions against hypervolume indicator**: the enhanced ϵ-constraint method obtained more unique feasible solutions, yielding in turn a Pareto front with a better hypervolume value (see Tables C.9 and C.10 and Figure C.17). Note however, that the hypervolume value does not depend exclusively on the number of Pareto solutions, but rather on their quality (i.e., the amount of space of the space that they cover). This is reflected in Tables C.9 and C.10, where the Sobol RS method shows more unique solutions than the Halton RS, yet the Halton RS yields a better hypervolume (Tables C.9 and C.10).

In summary, we find that in this case study, both strategies (i.e., reducing objectives and using pseudo-random sampling) improve the performance of the algorithm from the viewpoints of number of feasible solutions generated and quality of the Pareto front.

8.4.2 Sustainable planning of Hydrogen supply chains

This example deals with the optimal design of a hydrogen SC for vehicle use in Spain taking into account economic and environmental concerns. The problem, which was first proposed by Almansoori and Shah (2009),

considers different technologies for production, storage and transportation of hydrogen to be established in a set of geographical regions distributed all over the country. The goal is to determine the optimal network configuration in terms of its economic and environmental performance.

The problem can be formulated as a multi-objective MILP that seeks to minimize the total cost of the network and its environmental impact. In this formulation, integer variables indicate the number of plants and storage facilities to be opened in a specific region (i.e., grid), whereas binary variables are employed to denote the existence of transportation links connecting the SC entities. The environmental impact was quantified via 15 life cycle assessment indicators based on the Eco-indicator 99 methodology (Hischier et al., 2010). The objectives considered are taking account in this order: the cost, the acidification potencial, the climate change, the eutrophication potencial, the freshwater aquatic ecotoxicity, the freshwater sediment ecotoxicity, the human toxicity, the ionising radiation the land use, the malodours air, the marine aquatic ecotoxicity, the marine sediment ecotoxicity, the photochemical oxidation, the resources, the stratospheric ozone depletion and the terrestrial ecotoxicity (Hischier et al., 2010). Further details on this case study can be found in a previous publication (Sabio et al., 2010). The MILP contains $11,804$ continuous variables, $7,304$ discrete variables and $29,495$ equations.

We proceeded in the same manner as before (see Algorithms 21 and 22 and 24). That is, we first generated 100 Pareto solutions using a bi-criteria algorithm in the original search space. A total of 15 2-objective models were constructed by combining the cost and each environmental objective. With the aim of obtaining a representative approximation of the

Pareto frontier, each solution was generated choosing randomly a different 2-objective model until obtaining 100 solutions. The equidistant sampling was carried out by optimizing the cost against environmental objectives considering the 3 partitions for each objective. The 2 interior partition points of each objective j are labeled as point a_j and point b_j. According to the Algorithm 24, an initial word is chosen as seed, then from this seed (i.e $abb \dots ab$), then we solve the next 300 sub-problems in lexicographic order considering the environmental objectives in the order presented above. In this case study we only ran 500 iterations in contrast to the $2,000$ iterations we run in the previous case, since in this case each iteration requires high CPU time. The objective reduction method (Copado-Méndez et al., 2014) identified 13 redundant objectives, taking 4 seconds for this task. We next ran 300 iterations of the enhanced and standard ϵ-constraint methods using each sampling technique (random, Halton and Sobol). Figure C.18, which is equivalent to Figure C.17, summarizes the results obtained.

Table C.11 is equivalent to Table C.9 and Table C.12 is equivalent to Table C.10. It can be seen in Table C.11 and Figure C.18 that the best approach, according to hypervolume indicator obtained after completing the 300 iterations is the ϵ-constraint method with Sobol sampling run in the reduced space (RS) followed by the Sobol in the full space (FS), random in the RS, Halton in the RS, random in the FS, equidistant in the RS, Halton in the FS and finally equidistant in the FS. We next proceed to analyze the results in further detail.

- **Full space (FS) against reduced space (RS)**: in this case, during the first 500 seconds, all the RS approaches outperform the FS meth-

ods in terms of both feasible solutions and hypervolume, as seen in Table C.12. Besides, the RS approaches are able to perform the 300 iterations in a fraction of the CPU time required by the FS methods, as shown in Tables C.11 and C.12. This might be due to the fact that the RS methods shows less auxiliary constraints. However, when the 300 iterations have been completed, the RS approaches identify less unique feasible solutions than the corresponding FS ones as we can observe in Table C.11. This happens because the objective reduction in this case study is very strong, going from 16 to only 3 criteria which, when optimized, give rise to many repeated solutions thus reducing the number of unique feasible solutions. The only exception for this tendency is the ϵ-constraint with equidistant sampling, which in the FS was unable to obtain a single feasible solution. Note that running the FS equidistant method considering 3 partitions for every objective would lead to 2^{15} iterations out of which we only performed 300. Because of this, the probability to obtain a feasible solution in these 300 iterations is extremely low.

- **Equidistant sampling against pseudo/quasi-random sampling**: the ϵ-constraint with pseudo/quasi-random sampling outperforms the equidistant sampling in terms of hypervolume indicator and number of feasible solutions, as seen in Table C.12. The reason why this happens is the same as in the previous case, i.e, solvers firstly try to identify solutions involving active constraints so that when the epsilon parameters are chosen by pseudo/quasi-random sequences, it is more likely that there will be an integer solution satisfying the epsilon con-

straints as an equality and thus the solver will find this solution rather fast (Ehrgott and Ryan, 2003). On the other hand, when epsilon parameters are chosen by the equidistant method, it is more unlikely that a feasible integer solution will exist when the epsilon constraint is active, and thus the solver may waste several iterations until dismissing the possibility of finding a solution at that point Interestingly, in Table C.11, the Halton sequence in the FS is the lowest in terms of hypervolume, yet, it identifies the largest number of feasible solutions.

- **Pseudo-random against quasi-random sampling**: as seen in Table C.11, the Sobol sequence outperforms the pseudo-random method in both, number of unique solutions and hypervolume indicator. We already explained in the previous case study the advantages of quasi-random sequences over the pseudo-random sequence. Note, however, that the Halton sequence did not work so well in this case (i.e., it obtained the lowest hypervolume), while in the FS, it identified the highest number of feasible solutions. Recall that obtaining many solutions does not necessarily guarantee a better hypervolume value, as shown in Tables C.11 and C.12, where it can be seen the Halton FS identified more solutions than the other methods, but obtained the second lowest hypervolume value.

- **Number of feasible solutions against hypervolume indicator**: in the first 500 seconds, the enhanced ϵ-constraint with pseudo/quasi-random sampling in the RS obtained more unique feasible solutions which did lead to higher hypervolume values than the FS methods as seen in Table C.12. On the contrary, if more time is allowed so

that the 300 iterations can be completed, some pseudo/quasi-random approaches in the FS achieve more unique solutions than the corresponding RS approaches yet their hypervolume is still lower. Hence, there is no evidence that a higher number of solutions necessarily leads to a higher hypervolume. For instance the Halton sequence in the FS identifies the highest number of feasible solutions but leads to the second lowest hypervolume (see Table C.11).

Finally, we find that the best approach in terms of the hypervolume indicator obtained after 300 iterations is the Sobol in the RS, followed by the Sobol in the FS in terms of hypervolume indicator. If we took into account time requirements, we would then choose the pseudo-random or Halton methods in the RS instead of the Sobol in the FS. In this second case study, the performance patterns observed in the first case are not so well defined, probably due to numerical issues, yet in general the same tendencies still hold. In both cases, the use of objective reduction or/and pseudo/quasi-random sampling leads to significant improvements when compared to the standard ϵ-constraint method, mainly in terms of CPU time and quality of the Pareto set.

8.5 Conclusions and future work

This work introduced an enhanced ϵ-constraint method that incorporates two main ingredients: an objective reduction technique, and the use of pseudo/quasi-random sampling methods for generating the auxiliary epsilon parameter values. To test the capabilities of our approach, we applied it to the multi-objective optimization of supply chains considering both,

economic and environmental concerns. For this comparison, we used several sampling techniques (pseudo-random, Halton and Sobol and standard, that is, equidistant) combined with an objective reduction algorithm. The performance of each method was assessed in terms of number of unique feasible solutions and hypervolume indicator values.

The experiments reveal that the best sampling techniques are the quasi-random sequences (Halton or Sobol), since they improve the performance from the viewpoints of number of feasible solutions and hypervolume value per time unit. This is due to the property of low-discrepancy, which distributes the epsilon parameters values in a more efficient manner compared to the pseudo-random and equidistant techniques.

Concerning the objective reduction approach, the results show that it expedites the ϵ-constraint method by eliminating auxiliary epsilon constraints.

Finally, we conclude that combining quasi-random sampling methods with objective reduction improves in general the overall performance of the standalone ϵ-constraint method in terms of number of feasible solutions and hypervolume per time unit. Our enhanced method can find many applications in a wide variety of multi-objective problems arising in all domains of science and engineering.

8.6 Acknowledgments

Pedro J. Copado wishes to acknowledge support of this research work from the Spanish Ministry of Education and Science (DPI2008-04099/DPI). The authors wish also to acknowledge support from the Spanish Ministry of Edu-

cation and Science (Projects CTQ2012-37039-C02, DPI2012-37154-C02-02 and ENE2011-28269-C03-03).

Part IV

Additional Material

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

Appendices

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTION TO THE DEVELOPMENT OF EFFICIENT ALGORITHMS FOR SOLVING COMPLEX SINGLE-OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION MODELS.
Pedro Jesús Copado méndez
Dipòsit Legal: T 1773-2014

Appendix A

Models

A.1 Model Hydrogen

We provide next a brief overview of the mathematical models used in our work. Further details can be found in Sabio et al. (2010), Guillén-Gosálbez et al. (2010) and Kostin et al. (2011b).

Equation A.1 defines the mass balance for the grids considered in the analysis, whereas Equation A.2 forces the model to fulfill a minimum demand satisfaction level. Equation A.3 limits the production capacity between lower and upper bounds. Equation A.4 determines the production capacity in a time period from the previous one plus the expansion in capacity executed in the same period. Equation A.5 limits the capacity expansions within lower and upper bounds given by the number of facilities

opened.

$$\sum_{s\in SI(i)} S_{igst-1} + \sum_{p} PR_{igpt} + \sum_{g'\neq g}\sum_{l} Q_{ilg'glt} = \sum_{s\in SI(i)} S_{igst} + D_{igt} + \sum_{g'\neq g}\sum_{l} Q_{ilgg'lt} \forall i,g,t \tag{A.1}$$

$$\overline{D_{gt}} dsat \leq \sum_{i} D_{igt} \leq \overline{D_{gt}} \forall g,t \tag{A.2}$$

$$\tau C^{PL}_{gpt} \leq \sum_{i} PR_{igpt} \leq C^{PL}_{gpt} \forall g,p,t \tag{A.3}$$

$$C^{PL}_{gpt} = C^{PL}_{gpt-1} + CE^{PL}_{gpt} \quad \forall g,p,t \tag{A.4}$$

$$\underline{PC^{PL}_{p}} N^{PL}_{gpt} \leq CE^{PL}_{gpt} \leq \overline{PC^{PL}_{p}} N^{PL}_{gpt} \forall g,p,t \tag{A.5}$$

Equations A.6 to A.9 are equivalent to equations A.3 to A.5, but apply to warehouses. Particularly, equation A.6 limits the amount of materials stored to be lower than the existing capacity. Equation A.7 forces the average inventory level, which is determined from the demand and turnover ratio, to be lower than the existing capacity. Equation A.8 provides the storage capacity in a time period from the previous one and the expansion in capacity in the previous period, whereas equation A.9 limits the expansion in capacity between lower and upper limits given by the number of storage

facilities installed.

$$\sum_{i \in IS(s)} S_{igst} \leq C^{ST}_{gst} \forall g, s, t \tag{A.6}$$

$$2(\theta D_{igt}) \leq \sum_{s \in SI(i)} C^{ST}_{gst} \forall i, g, t \tag{A.7}$$

$$C^{ST}_{gst} = C^{ST}_{gst-1} + CE^{ST}_{gst} \forall g, s, t \tag{A.8}$$

$$\underline{SC^{ST}_{s}} N^{ST}_{gst} \leq CE^{ST}_{gst} \leq \overline{SC^{ST}_{s}} N^{ST}_{gst} \forall g, s, t \tag{A.9}$$

Equation A.10 limits the transportation links between lower and upper bounds provided the link is finally established. Equations A.11 and A.12 are defined for the construction of pipelines. Equation A.13 is a logic constraint that makes the formulation tighter. Equations A.14 and A.15 avoid the transportation between certain maritime grids, whereas equation A.16 is a symetric cut. Finally, equations A.17 to A.31 allow to determine the cost of the network.

$$\begin{aligned} &\underline{QC_{lgg'}} X_{gg'lt} \leq \sum_{i} Q_{ilgg't} \leq \overline{QC_{lgg'}} X_{gg'lt} \\ &\forall g, g' (g \neq g'), l \in LI(i) \cup NPL, t \end{aligned} \tag{A.10}$$

$$\sum_{t' \leq t+1} \underline{QC_{lgg'}} X_{gg'lt'} \leq \sum_{i} Q_{ilgg't} \leq \sum_{t' \leq t+1} \overline{QC_{lgg'}} X_{gg'lt} \quad \forall g, g'(g \neq g'), l = pipeline, t \tag{A.11}$$

$$\sum_{t' \leq t+1} X_{gg'lt'} \leq 1 \qquad \forall g, g'(g \neq g'), l = pipeline, t \tag{A.12}$$

$$X_{gg'lt} + X_{g'glt} \leq 1 \qquad \forall g, g'(g \neq g'), l \in LI(i,t \tag{A.13}$$

$$X_{lgg't} = 0 \qquad \forall l, g, g' \in LG' \quad LG' = \{l, g, g' : (l = ship) \wedge ((g, g') \notin SGG(gg'))\} \tag{A.14}$$

$$X_{lgg't} = 0 \qquad \forall l, g, g' \in LG \quad LG = \{l, g, g' : (l \neq ship) \wedge ((g, g') \in SGG'(gg'))\} \tag{A.15}$$

$$X_{lgg't} = 0 \qquad \forall l, g = g' \tag{A.16}$$

$$TDC = \sum_{t} \frac{TC}{(1+ir)^{t-1}} \tag{A.17}$$

$$TC_t = FCC_t + TCC_t + FOC_t + TOC_t \qquad \forall t \tag{A.18}$$

$$FOC_t = \sum_i \sum_g \sum_p upc_{igpt} PR_{igpt} + \sum_i \sum_g \sum_s \in SI(i) usc_{igst} \left(\theta D_{igt}\right) \qquad \forall t \tag{A.19}$$

$$FCC_t = \sum_g \sum_p \left(\alpha_{gpt}^{PL} N_{gpt}^{PL} + \beta_{gpt}^{PL} CE_{gpt}^{PL}\right) + \sum_g \sum_s \left(\alpha_{gst}^{ST} N_{gst}^{ST} + \beta_{gst}^{ST} CE_{gst}^{ST}\right) \qquad \forall t \tag{A.20}$$

$$TCC_t = \sum_{l \neq ship, pipeline} N_{lt}^{TR} \cdot cc_{lt} + PCC_t \tag{A.21}$$

$$PCC(t) = \sum_g \sum_{g' \neq g} \sum_{l \in LI(i)} upcc_t X_{lgg't} distance_{gg'} \qquad \forall t \tag{A.22}$$

$$\sum_{t' \leq t+1} N_{lt'}^{TR} \geq \sum_{i \in IL(l)} \sum_g \sum_{g' \neq g} \sum_t \frac{Q_{igg'lt}}{av_l tcap_l} \left(\frac{2distance_{gg'}}{speed_l} + lutime_l\right)$$

$$\forall l \neq ship, pipeline \tag{A.23}$$

$$TOC_t = ROC_t + POC_t + SOC_t \qquad \forall t \tag{A.24}$$

$$ROC_t = FC_t + LC_t + MC_t + GC_t \qquad \forall t \tag{A.25}$$

$$FC_t = \sum_i \sum_g \sum_{g' \neq g} \sum_{l \in LI(i)} fuelp_{lt} \frac{2distance_{gg'} Q_{ilgg't}}{fuelc_l tcap_l} \qquad \forall t \tag{A.26}$$

$$\begin{aligned} LC_t = & \sum_i \sum_g \sum_{g' \neq g} \sum_{l \in LI(i)} wage_{lt} \\ & \times \left[\frac{Q_{ilgg't}}{tcap_l} \left(\frac{2distance_{gg'}}{speed_l} + lutime_l \right) \right] \qquad \forall t \end{aligned} \tag{A.27}$$

$$MC_t = \sum_i \sum_g \sum_{g' \neq g} \sum_{l \in LI(i)} cud_l \frac{2distance_{gg'} Q_{ilgg't}}{tcap_l} \qquad \forall t \tag{A.28}$$

$$GC_t = \sum_l \sum_{t' \leq t} g_{lt} N_{lt'}^{TR} \qquad \forall t \tag{A.29}$$

$$POC(t) = \sum_i \sum_g \sum_{g' \neq g} \sum_{l \in LI(i)} upoc_t Q_{ilgg't} \qquad \forall t \tag{A.30}$$

$$SOC_t = \sum_i \sum_g \sum_{g' \neq g} \sum_{l \in LI(i)} usoc_t \left(\frac{distance_{gg'}}{speed_l} \right) Q_{ilgg't} \qquad \forall t \tag{A.31}$$

A.1.1 Notation

Indices

e	scenarios
i	hydrogen form
g	potential locations
l	transportation mode
p	manufacturing technologies
s	storage technologies
t	time period

Sets

$IL(l)$	set of hydrogen forms that can be transported via transportation mode l
$IS(s)$	set of hydrogen forms that can be stored via technology s
$LI(i)$	set of transportation modes that can transport hydrogen form i
$SI(i)$	set of storage technologies that can store hydrogen form i

Parameters

av_l	availability of transportation mode l
cc_{lt}	capital cost of transport mode l in period t
cud_{lt}	maintenance cost of transportation mode l in period t per unit of distance traveled
$\overline{D_{gt}}$	total demand of hydrogen in location g in period t
$distance_{gg'}$	average distance traveled between locations g and g'
$dsat$	demand satisfaction level to be fulfilled
$fuelc_l$	fuel consumption of transportation mode l
$fuelp_{lt}$	price of the fuel consumed by transportation mode l in period t
ge_{lt}	general expenses of transportation mode l in period t
ir	interest rate
$lutime_l$	loading/unloading time of transportation mode l
$\overline{PC_p^{PL}}$	upper bound on the capacity expansion of manufacturing technology p
$\underline{PC_p^{PL}}$	lower bound on the capacity expansion of manufacturing technology p
$\overline{QC_{gg'l}}$	upper bound on the flow of materials between locations g and g' via transportation model l
$\underline{QC_{gg'l}}$	lower bound on the flow of materials between locations g and g' via transportation model l
$\overline{SC_s^{ST}}$	upper bound on the capacity expansion of storage technology s
$\underline{SC_s^{ST}}$	lower bound on the capacity expansion of storage technology s
$speed_l$	average speed of transportat mode l

$tcap_l$	capacity of transport mode l
upc_{igpte}	mean value of unit production cost of hydrogen form i produced via technology p in location g in period t in scenario e
$Vupc_{igpte}$	Variance associated to the probability distribution of upc_{igpte}
usc_{igst}	unit storage cost of hydrogen form i stored via technology s in location g in period t
$wage_{lt}$	driver wage of transportation mode l in period t
α_{gpt}^{PL}	fixed investment term associated with manufacturing technology p installed in location g in period t
α_{gst}^{ST}	fixed investment term associated with storage technology s installed in location g in period t
β_{gpt}^{PL}	variable investment term associated with manufacturing technology p installed in location g in period t
β_{gst}^{ST}	variable investment term associated with storage technology s installed in location g in period t
θ	average storage period
τ	minimum desired percentage of the capacity that must be used
$prob_e$	occurrence probability of scenario e

Variables

C_{gpt}^{PL}	capacity of manufacturing technology p in location g in period t

C_{gst}^{ST}	capacity of storage technology s in location g in period t
CE_{gpt}^{PL}	capacity expansion of manufacturing technology p in location g in period t
CE_{gst}^{ST}	capacity expansion of storage technology s in location g in period t
D_{igt}	amount of hydrogen form i distributed in location g in period t
FC_t	fuel cost in period t
FCC_t	facility capital cost in period t
FOC_{te}	facility operating cost in period t in scenario e
GC_t	general cost in period t
LC_t	labor cost in period t
MC_t	maintenance cost in period t
$TPIC$	capital cost of pipelines establishment (euros/km)
UTP	unit transportation cost of pipelines (euros/kg day)
$UTCB$	unit transportation cost of ship rental (euros/h kg)
$PICC_t$	pipeline capital cost (euros/yr)
PIC_t	pipeline operating cost (euros/yr)
$TOCB_t$	ship operating cost
N_{gpt}^{PL}	number of plants of type p installed in location g in period t (integer variable)
N_{gst}^{ST}	number of storage facilities of type s installed in location g in period t (integer variable)
N_{lt}^{TR}	number of transportation units of type l purchased in period t (integer variable)

PR_{igpt}	production of hydrogen mode i via technology p in period t in location g
$Q_{igg'lt}$	flow of hydrogen mode i via transportation mode l between locations g and g' in period t
S_{igst}	amount of hydrogen in physical form i stored via technology s in location g in period t
TC_{te}	total amount of money spent in period t for scenario e
TCC_t	total transportation capital cost in period t
TDC_e	total discounted cost for scenario e
TOC_t	transportation operating cost in period t
$X_{gg'lt}$	binary variable (1 if a link between locations g and g' using transportation technology l is established, 0 otherwise)

A.2 Model Ethanol

Equation A.32 defines the mass balance for every grid and time period, whereas equation A.33 forces the sales to be lower than the demand. Equation A.34 determines the total production rate in a give grid, whereas equation A.35 provides the amount produced in a facility from the production rate of its main product. Equation A.36 limits the production rate within lower and upper bounds, whereas equation A.37 determines the production capacity from the previous one and the expansion in capacity in the current period. The expansions in capacity are given by the number of facilities opened, as expressed in constraint A.38. Constraint A.39 limits the amoung

of raw materials purchased.

$$\sum_{s\in SI(i)} ST_{isgt-1} + PT_{igt} + PU_{igt} + \sum_{l\in LI(i)}\sum_{g'\neq g} Q_{ilg'gt} = \sum_{s\in SI(i)} ST_{isgt} + DTS_{igt} + \sum_{l\in LI(i)}\sum_{g'\neq g} Q_{ilgg't} + W_{igt} \qquad \forall i,g,t \tag{A.32}$$

$$DTS_{igt} \leq SD_{igt} \qquad \forall i,g,t \tag{A.33}$$

$$PT_{igt} = \sum_{p} PE_{ipgt} \qquad \forall i,g,t \tag{A.34}$$

$$PE_{ipgt} = \rho_{pi} PE_{i'pgt} \qquad \forall i,p,g,t \qquad \forall i' \in IM(p) \tag{A.35}$$

$$\tau PCap_{pgt} \leq PE_{ipgt} \leq PCap_{pgt} \qquad \forall i,p,g,t \tag{A.36}$$

$$PCap_{pgt} = PCap_{pgt-1} + PCapE_{pgt} \qquad \forall p,g,t \tag{A.37}$$

$$\underline{PCap_p} NP_{pgt} \leq PCapE_{pgt} \leq \overline{PCap_p} NP_{pgt} \qquad \forall p,g,t \tag{A.38}$$

$$PU_{igt} \leq CapCrop_{gt} \qquad \forall i = Sugar\ cane, g, t \tag{A.39}$$

$$\underline{SCap_s} NS_{sgt} \leq SCapE_{sgt} \leq \overline{SCap_s} NS_{sgt} \qquad \forall s, g, t \tag{A.40}$$

Constraint A.40 limits the expansions in capacity of the storage facilities within lower and upper bounds given by the number of facilities opened. Equation A.41 provides the storage capacity of a region from the previous one and the expansion in capacity in the current period. Equation A.42 constraints the amount of materials stored in a region to be lower than the installed capacity in that region. Constraint A.43 determines the average inventory level from the product sales, whereas equation A.44 forces this average inventory to be lower than the existing capacity.

$$SCap_{sgt} = SCap_{sgt-1} + SCapE_{sgt} \qquad \forall s, g, t \tag{A.41}$$

$$\sum_{i \in IS(s)} ST_{isgt} \leq SCap_{sgt} \qquad \forall s, g, t \tag{A.42}$$

$$AIL_{igt} = \beta DTS_{igt} \qquad \forall i, g, t \tag{A.43}$$

$$2AIL_{igt} \leq \sum_{s \in SI(i)} SCap_{sgt} \qquad \forall i, g, t \tag{A.44}$$

Constraint A.45 models the establishment of transportation links between two grids, whereas equation A.46 is a logic cut that makes the formulation tighter.

$$\underline{Q_l} X_{lgg't} \leq \sum_{i \in IL(l)} Q_{ilgg't} \leq \overline{Q_l} X_{lgg't} \qquad \forall l, t, g, g' (g' \neq g) \tag{A.45}$$

$$X_{lgg't} + X_{lg'gt} = 1 \qquad \forall l, t, g, g'(g' \neq g) \tag{A.46}$$

Equations A.47 to A.64 are added to determine the NPV of the network.

$$ROI = \frac{(\sum_t CF_t)/T}{FCI} \tag{A.47}$$

$$NPV = \sum_t \frac{CF_t}{(1+ir)^{t-1}} \tag{A.48}$$

$$CF_t = NE_t - FTDC_t \qquad t = 1, ..., T-1 \tag{A.49}$$

$$CF_t = NE_t - FTDC_t + svFCI \qquad t = T \tag{A.50}$$

$$NE_t = (1-\varphi)(Rev_t - FOC_t - TOC_t) + \varphi DEP_t \qquad \forall t \tag{A.51}$$

$$Rev_t = \sum_{i \in SEP} \sum_g DTS_{igt} PR_{igt} \qquad \forall t \tag{A.52}$$

$$\begin{aligned} FOC_t = & \sum_i \sum_g \sum_{p \in IM(p)} UPC_{ipgt} PE_{ipgt} + \\ & \sum_i \sum_g \sum_{s \in IS(s)} USC_{isgt} AIL_{igt} + \\ & DC_t \qquad \forall t \end{aligned} \tag{A.53}$$

$$DC_t = \sum_i \sum_g W_{igt} LT_{ig} \qquad \forall t \tag{A.54}$$

$$TOC_t = FC_t + LC_t + MC_t + GC_t \qquad \forall t \tag{A.55}$$

$$FC_t = \sum_g \sum_{g' \neq g} \sum_l \sum_{i \in IL(l)} \left[\frac{2EL_{gg'}\ Q_{ilgg't}}{FE_l\ TCap_l} \right] FP_{lt} \qquad \forall t \tag{A.56}$$

$$LC_t = \sum_g \sum_{g' \neq g} \sum_l DW_{lt} \sum_{i \in IL(l)} \left[\frac{Q_{ilgg't}}{TCap_l} \left(\frac{2EL_{gg'}}{SP_l} + LUT_l \right) \right] \qquad \forall t \tag{A.57}$$

$$MC_t = \sum_g \sum_{g' \neq g} \sum_l \sum_{i \in IL(l)} ME_l \frac{2EL_{gg'} Q_{ilgg't}}{TCap_l} \qquad \forall t \tag{A.58}$$

$$GC_t = \sum_l \sum_{t' \leq t} GE_{lt} NT_{lt'} \qquad \forall t \tag{A.59}$$

$$DEP_t = \frac{(1 - sv) FCI}{T} \qquad \forall t \tag{A.60}$$

$$\begin{aligned} FCI = & \sum_p \sum_g \sum_t (\alpha^{PL}_{pgt} NP_{pgt} + \beta^{PL}_{pgt} PCapE_{pgt}) + \\ & \sum_s \sum_g \sum_t (\alpha^{S}_{sgt} NS_{sgt} + \beta^{S}_{sgt} SCapE_{sgt}) + \\ & \sum_l \sum_t (NT_{lt} TMC_{lt}) \end{aligned} \tag{A.61}$$

$$\sum_{t \leq T} NT_{lt} \geq \sum_{i \in IL(l)} \sum_{g} \sum_{g' \neq g} \sum_{t} \frac{Q_{ilgg't}}{avl_l TCap_l} \left(\frac{2EL_{gg'}}{SP_l} + LUT_l \right) \qquad \forall l \quad \text{(A.62)}$$

$$FCI \leq \overline{FCI} \qquad \text{(A.63)}$$

$$FTDC_t = \frac{FCI}{T} \qquad \forall t \qquad \text{(A.64)}$$

A.2.1 Notation

Indices

i	materials
g	sub-region zones
l	transportation modes
p	manufacturing technologies
s	storage technologies
t	time periods

Sets

$IL(l)$	set of materials that can be transported via transportation mode l
$IM(p)$	set of main products for each technology p
$IS(s)$	set of materials that can be stored via storage technology s
SEP	set of products that can be sold
$SI(i)$	set of storage technologies that can store materials i

Parameters

α^{PL}_{pgt}	fixed investment coefficient for technology p
α^{S}_{sgt}	fixed investment coefficient for storage technology s
β	storage period
β^{PL}_{pgt}	variable investment coefficient for technology p
β^{S}_{sgt}	variable investment coefficient for storage technology s
ρ_{pi}	material balance coefficient of material i in technology p
τ	minimum desired percentage of the available installed capacity
φ	tax rate
avl_l	availability of transportation mode l
$CapCrop_{gt}$	total capacity of sugar cane plantations in sub-region g in time t
DW_{lt}	driver wage
$EL_{gg'}$	distance between g and g'
$\overline{FCI}$	upper limit for capital investment
FE_l	fuel consumption of transport mode l
FP_{lt}	fuel price

GE_{lt}	general expenses of transportation mode l
LT_{ig}	landfill tax
ME_l	maintenance expenses of transportation mode l
$\overline{PCap_p}$	maximum capacity of technology p
$\underline{PCap_p}$	minimum capacity of technology p
PR_{igt}	prices of final products
$\overline{Q_l}$	maximum capacity of transportation mode l
$\underline{Q_l}$	minimum capacity of transportation mode l
$\overline{SCap_s}$	maximum capacity of technology p
$\underline{SCap_s}$	minimum capacity of storage technology s
SD_{igt}	actual demand of product i in sub-region g in time t
SP_l	average speed of transportation mode l
sv	salvage value
T	number of time intervals
$TCap_l$	capacity of transportation mode l
TMC_{lt}	cost of establishing transportation mode l in period t
UPC_{ipgt}	unit production cost
USC_{isgt}	unit storage cost

Variables

CF_t	cash flow in time t
DC_t	disposal cost in time t
DTS_{igt}	delivered amount of material i in sub-region g in period t
FC_t	fuel cost
FCI	fixed capital investment

FOC_t	facility operating cost in time t
$FTDC_t$	fraction of the total depreciable capital in time t
GC_t	general cost
LC_t	labor cost
MC_t	maintenance cost
NE_t	net earnings in time t
NP_{pgt}	number of installed plants with technology p in sub-region g in time t
NPV	net present value of SC
NS_{sgt}	number of installed storages with storage technology s in sub-region g in time t
NT_{lt}	number of transportation units l
$PCap_{pgt}$	existing capacity of technology p in sub-region g in time t
$PCapE_{pgt}$	expansion of the existing capacity of technology p in sub-region g in time t
$Q_{ilgg't}$	flow rate of material i transported by mode l from sub-region g' to current sub-region g in time period t
Rev_t	revenue in time t
RNP_{pgt}	"relaxed" number of installed plants with technology p in sub-region g in time interval t
RNS_{sgt}	"relaxed" number of installed storages with storage technology s in sub-region g in time interval t
RNT_{lt}	"relaxed" number of transportation units l in time interval t
$SCap_{sgt}$	capacity of storage s in sub-region g in time t

$SCapE_{sgt}$	expansion of the existing capacity of storage s in sub-region g in time t
ST_{isgt}	total inventory of material i in sub-region g stored by technology s in time t
TOC_t	transport operating cost in time t
PE_{ipgt}	production rate of material i in technology p in sub-region g in time t
PT_{igt}	total production rate of material i in sub-region g in time t
PU_{igt}	purchase of material i in sub-region g in time t
$X_{lgg't}$	binary variable, which is equal to 1 if material flow between two sub-regions g and g' is established and 0 otherwise
W_{igt}	amount of wastes i generated in sub-region g in period t

Appendix B

Proofs and Normalization

B.1 Proof of Theorem B.1.1

Theorem B.1.1 shows that the solution of each sub-problem MOR_{μ_p} solved at iteration p is a relaxation of the full space MILP, and it in turn provides a rigorous lower bound on the global optimum of the full space MILP. We demonstrate this property below:

Theorem B.1.1 *Let δ_{μ_p} be the optimal solution of sub-problem MOR_{μ_p} solved at iteration p and size μ of the algorithm (which is defined for the subset of solutions S_{μ_p} contained in S), and b_j be a parameter used in cut j that corresponds to the optimal solution of an instance of problem MOR that is used to construct cut j (i.e., instance MOR_j defined for solutions S_j). The inequalities from 7.12 to 7.15 are a valid cut for problem MOR_{μ_p}.*

Proof The proof is by contradiction. We claim that inequalities 7.12 and 7.15 do not chop off any feasible solution of problem MOR. Assume that there is a feasible solution of MOR such that for this solution (i.e., combination of objectives omitted) the value of the auxiliary variable a_j (denoted by $\overline{a_j}$) is strictly lower than b_j. From equations 7.13 to 7.15, it follows that

for such combination of objectives omitted, the value of the error (denoted by $\overline{\delta(s, s', i)}$) in every objective and pair of solutions in the set S_j will be strictly lower than b_j, that is:

$$\overline{\delta(s, s', i)} < b_j \quad \forall s, s' \in S_j, \quad \forall j \in CUT_{\mu_p}, \quad \forall i \tag{B.1}$$

This contradicts the fact that b_j is the global optimum of problem MOR_j ■.

B.2 Normalization of the Pareto optimal solutions

A normalization step is applied to the Pareto set of solutions in order to make them comparable in all of the objectives. Several methods are available for this purpose (Cloquell V, Santamarina M, 2001). We have used in our case the following expression:

$$v'_i = \frac{v_i - v_{MIN}}{v_{MAX} - v_{MIN}}$$

where v'_i is the normalized value, and v_i is the original value. This method covers exactly the range $[0, 1]$, where zero is the best value and one the worst value (we consider that we aim to minimize all of the objectives simultaneously).

Appendix C

Tables and Figures

C.1 Tables

Per	Con	Bin	Int	Dec	Tot	Equ
2	$1.03x10^4$	$4.57x10^4$	$4.80x10^2$	$4.62x10^4$	$5.65x10^4$	$1.07x10^5$
4	$2.06x10^4$	$9.15x10^4$	$9.60x10^2$	$9.24x10^4$	$1.13x10^5$	$2.14x10^5$
6	$3.09x10^4$	$1.37x10^5$	$1.40x10^3$	$1.39x10^5$	$1.70x10^5$	$3.21x10^5$
8	$4.18x10^4$	$1.84x10^5$	$1.92x10^3$	$1.85x10^5$	$2.26x10^5$	$4.28x10^5$
10	$5.15x10^4$	$2.29x10^5$	$2.4x10^3$	$2.31x10^5$	$2.83x10^5$	$5.35x10^5$
12	$6.17x10^4$	$2.74x10^5$	$2.88x10^3$	$2.77x10^5$	$3.39x10^5$	$6.42x10^5$
14	$7.20x10^4$	$3.20x10^5$	$3.36x10^3$	$3.24x10^5$	$3.96x10^5$	$7.49x10^5$
16	$8.23x10^4$	$3.66x10^5$	$3.84x10^3$	$3.70x10^5$	$4.52x10^5$	$8.56x10^5$

Table C.1: Number of variables and equations of hydrogen. For each period, we display number of continuous variables, number of binary variables, number of integer variables, number of decision variables (Bin+Int), total number of variables, and finally number of equations.

Per	Time	Bst Time	Bst Cost	Avg Time	Std Time	Avg Cost	Std Cost
2	$1.00x10^3$	$7.2.8x10$	$1.05x10^{12}$	$4.78x10^2$	$2.48x10^2$	$1.05x10^{12}$	$1.89x10^7$
4	$2.00x10^3$	$1.39x10^2$	$1.30x10^{12}$	$7.48x10^2$	$5.81x10^2$	$1.30x10^{12}$	$3.80x10^7$
6	$3.00x10^3$	$2.73x10^2$	$1.54x10^{12}$	$1.16x10^3$	$7.84x10^2$	$1.54x10^{12}$	$4.20x10^7$
8	$4.00x10^3$	$2.95x10^2$	$1.79x10^{12}$	$1.65x10^3$	$1.20x10^3$	$1.79x10^{12}$	$5.83x10^7$
10	$5.00x10^3$	$2.96x10^2$	$2.02x10^{12}$	$2.75x10^3$	$1.66x10^3$	$2.02x10^{12}$	$1.12x10^8$
12	$6.00x10^3$	$6.39x10^2$	$2.24x10^{12}$	$2.97x10^3$	$1.51x10^3$	$2.24x10^{12}$	$1.29x10^8$
14	$7.00x10^3$	$5.80x10$	$2.45x10^{12}$	$1.95x10^2$	$1.26x10^2$	$2.45x10^{12}$	0
16	$8.00x10^3$	$1.45x10^2$	$2.64x10^{12}$	$2.28x10^2$	$1.17x10^2$	$2.64x10^{12}$	0

Table C.2: Results of the LNS algorithm to the problem of HYDROGEN. The notation used in the table is as follows: Per stands for time periods, Bst is the time at which the best solution calculated by the LNS has been found, Bst Cost is the best objective function value in all the runs, Avg Time and Avg Cost represent the average objective function value and CPU time, respectively, of the LNS algorithm over all the replications executed, while Std Time and Std Cost are the standard deviation of the objective function and CPU time, respectively. Note that in periods 14 and 16 the standard deviation of the CPU time is zero because the algorithm convergence is very fast.

Per	CPLEX 12h	CPLEX Time	LNS Avg	LNS Bst
2	0.05	0.05	0.05	0.05
4	0.05	0.06	0.07	0.06
6	0.06	0.07	0.07	0.07
8	0.08	Not available	0.08	0.08
10	0.10	Not available	0.09	0.09
12	Not available	Not available	Not available	Not available
14	Not available	Not available	Not available	Not available
16	Not available	Not available	Not available	Not available

Table C.3: Hydrogen GAP's. GAP's are calculated respect to lower bound found by CPLEX for 12h. This table displays: the best solution calculated by CPLEX after 12 hours of CPU time and after the same CPU time provided to the LNS, the best solution found by the LNS and the average solution calculated by the LNS. Note that in some instances, CPLEX is unable to provide any bound even after the aforementioned CPU time. No result means that CPLEX was unable to provide even a feasible solution after the specified CPU time. Not available means that the quality of the solution produced by the LNS cannot be determined since CPLEX did not provide any reference solution for comparison purposes after the specified running time.

Per	**Con**	**Bin**	**Int**	**Dec**	**Tot**	**Equ**
2	$3.72x10^4$	$3.46x10^3$	$3.42x10^2$	$3.80x10^3$	$4.10x10^4$	$2.18x10^4$
4	$7.42x10^4$	$6.91x10^3$	$6.84x10^2$	$7.60x10^3$	$8.18x10^4$	$4.34x10^4$
6	$1.11x10^5$	$1.04x10^4$	$1.03x10^3$	$1.14x10^4$	$1.23x10^5$	$6.50x10^4$
8	$1.48x10^5$	$1.38x10^4$	$1.37x10^3$	$1.52x10^4$	$1.63x10^5$	$8.66x10^4$
10	$1.85x10^5$	$1.73x10^4$	$1.71x10^3$	$1.90x10^4$	$2.04x10^5$	$1.08x10^5$
12	$2.22x10^5$	$2.07x10^4$	$2.05x10^3$	$2.28x10^4$	$2.45x10^5$	$1.30x10^5$
14	$2.59x10^5$	$2.42x10^4$	$2.39x10^3$	$2.66x10^4$	$2.86x10^5$	$1.51x10^5$
16	$2.96x10^5$	$2.76x10^4$	$2.74x10^3$	$3.04x10^4$	$3.27x10^5$	$1.73x10^5$

Table C.4: Number of variables and equations of ethanol.

Per	Time	Bst Time	Bst Profit	Avg Time	Std Time	Avg Profit	Std Profit
2	$5.00x10^2$	$1.51x10^2$	$3.47x10^8$	$2.48x10^2$	$1.18x10^2$	$3.45x10^8$	$2.69x10^6$
4	$2.00x10^3$	$4.07x10^2$	$1.08x10^9$	$6.87x10^2$	$3.03x10^2$	$1.08x10^9$	0
6	$3.50x10^3$	$1.30x10^3$	$1.78x10^9$	$1.56x10^3$	$7.65x10^2$	$1.78x10^9$	$1.76x10^6$
8	$5.00x10^3$	$4.65x10^3$	$2.36x10^9$	$3.91x10^3$	$1.05x10^3$	$2.36x10^9$	$1.89x10^6$
10	$6.50x10^3$	$1.71x10^3$	$2.77x10^9$	$5.21x10^3$	$1.46x10^3$	$2.77x10^9$	$5.45x10^6$
12	$8.00x10^3$	$6.86x10^3$	$3.18x10^9$	$7.15x10^3$	$9.75x10^2$	$3.18x10^9$	$2.33x10^3$
14	$9.50x10^3$	$9.53x10^3$	$3.54x10^9$	$8.45x10^3$	$1.31x10^3$	$3.54x10^9$	$5.11x10^6$
16	$1.10x10^4$	$9.53x10^3$	$3.81x10^9$	$1.01x10^4$	$8.93x10^2$	$3.81x10^9$	$7.60x10^6$

Table C.5: Results of the LNS algorithm to the problem of ethanol

Per	CPLEX 12h	CPLEX Time	LNS Avg	LNS Bst
2	0.00	0.00	5.67	5.04
4	0.00	0.00	2.13	2.13
6	0.35	0.35	1.55	1.48
8	0.83	1.08	1.82	1.80
10	1.38	1.60	2.30	2.21
12	2.09	3.43	2.74	2.74
14	1.92	3.10	2.80	2.73
16	2.06	3.32	2.87	2.70

Table C.6: Ethanol GAP's. GAP's are calculated respect to upper bound found by CPLEX for 12h

Obj	MILP FS	Cut-MOSS		Exhaustive Method	Greedy	
Removed	CPU times(s)	Iterations	CPU times(s)	CPU times(s)	CPU times(s)	Optimality Gap (%)
1	$8.33x10^3$	1	7.22	$1.78x10^3$	$1.00x10^{-5}$	0
2	$8.23x10^3$	1	7.31	$1.74x10^3$	$1.34x10^{-5}$	0
3	$8.42x10^3$	1	7.44	$1.86x10^3$	$2.45x10^{-5}$	0
4	$8.12x10^3$	1	7.46	$1.86x10^3$	$3.65x10^{-5}$	0
5	$8.20x10^3$	1	7.36	$1.86x10^3$	$6.70x10^{-5}$	0
6	$8.20x10^3$	1	7.39	$1.87x10^3$	$8.23x10^{-5}$	0
7	$8.22x10^3$	1	7.40	$1.86x10^3$	$2.13x10^{-5}$	0
8	$8.33x10^3$	1	7.61	$1.86x10^3$	$9.83x10^{-4}$	0
9	$8.13x10^3$	1	7.52	$2.17x10^3$	$2.31x10^{-4}$	0
10	$8.92x10^3$	1	7.58	$1.96x10^3$	$3.10x10^{-4}$	0
11	$8.12x10^3$	1	7.75	$2.13x10^3$	$8.11x10^{-4}$	0
12	$8.34x10^3$	8	$5.62x10$	$1.20x10^3$	$1.32x10^{-4}$	0
13	$8.12x10^3$	6	$1.25x10^2$	$4.46x10^2$	$2.40x10^{-4}$	0
14	$8.22x10^3$	8	$1.13x10^2$	$9.50x10$	$5.65x10^{-4}$	0
15	$8.43x10^3$	8	$6.12x10$	$3.05x10$	$1.11x10^{-4}$	0

Table C.7: Comparison for the hydrogen SC: MILP FS (full-space MILP), Cut-MOSS (our algorithm), exhaustive method and greedy algorithm Brockhoff and Zitzler (2006a) for 16 objectives removing 1 to 15. The CPU time associated with the Cut-MOSS includes the time spent in generating the cuts. The time displayed for every algorithm considers a 0% optimality gap, except for the greedy method which offers no guarantee of global optimality. The greedy gap is obtained from the solution provided by the greedy algorithm and the global optimal solution determined by the exact methods.

Obj	MILP FS	Cut-MOSS		Exhaustive Method	Greedy	
Removed	CPU times(s)	Iterations	CPU times(s)	CPU times(s)	CPU times(s)	Optimality Gap (%)
1	$8.33x10^3$	1	7.55	$1.58x10^3$	$4.84x10^{-5}$	0
2	$8.19x10^3$	1	7.32	$1.58x10^3$	$2.92x10^{-5}$	0.91
3	$8.18x10^3$	1	7.44	$1.57x10^3$	$8.84x10^{-5}$	0.91
4	$8.49x10^3$	1	7.46	$1.59x10^3$	$9.44x10^{-5}$	0.91
5	$8.51x10^3$	1	7.36	$1.50x10^3$	$2.66x10^{-5}$	0.91
6	$8.20x10^3$	1	7.40	$1.50x10^3$	$1.90x10^{-5}$	0.91
7	$7.62x10^3$	1	7.40	$1.50x10^3$	$8.87x10^{-5}$	0.91
8	$8.33x10^3$	11	7.61	$1.50x10^3$	$4.62x10^{-5}$	0.91
9	$7.65x10^3$	1	7.52	$1.58x10^3$	$5.49x10^{-5}$	0.91
10	$7.37x10^3$	1	7.58	$1.48x10^3$	$9.28x10^{-5}$	0.91
11	$7.32x10^3$	9	7.75	$1.17x10^3$	$1.60x10^{-4}$	0.91
12	No solution	13	$4.69x10^2$	$5.30x10^2$	$9.81x10^{-4}$	0.91
13	No solution	12	$1.28x10^2$	$2.00x10^2$	$5.86x10^{-4}$	0.90
14	No solution	5	$1.13x10^2$	4.56	$2.24x10^{-4}$	0.74

Table C.8: Comparison for the metabolic network problem: MILP FS (full space MILP), Cut-MOSS (our algorithm), exhaustive method and greedy algorithms (Brockhoff and Zitzler, 2006a) for 14 objectives removing 1 to 14. The CPU time associated with the Cut-MOSS includes the time spent in generating the cuts. The time displayed for every algorithm considers a 0% optimality gap, except for the greedy method which offers no guarantee of global optimality. The greedy gap is obtained from the solution provided by the greedy algorithm and the global optimal solution determined by the exact methods.

2,000 Iterations	# Solutions	Total Time (sec)	Hypervolume
Halton RS	936	$5.45x10^3$	0.16
Sobol RS	943	$5.43x10^3$	0.15
Random RS	925	$5.40x10^3$	0.15
Sobol FS	557	$4.93x10^3$	0.05
Halton FS	531	$4.58x10^3$	0.05
Equidistant RS	127	$9.74x10^3$	0.05
Random FS	537	$4.61x10^3$	0.04
Equidistant FS	69	$12.43x10^3$	0.03

Table C.9: Final results obtained for each approach in Ethanol case.

2,000 seconds	# Solutions	Hypervolume
Halton RS	340	0.15
Sobol RS	347	0.14
Random RS	341	0.14
Sobol FS	233	0.05
Halton FS	222	0.05
Equidistant RS	48	0.05
Random FS	232	0.04
Equidistant FS	38	0.03

Table C.10: Intermediate results computed up to 2,000 seconds in Ethanol case

300 Iterations	# Solutions	Total Time (sec)	Hypervolume
Sobol RS	48	$10.60x10^2$	0.955
Sobol FS	203	$533.00x10^2$	0.948
Random RS	59	$16.80x10^2$	0.946
Halton RS	53	$12.30x10^2$	0.946
Random FS	204	$529.00x10^2$	0.944
Equidistant RS	24	$5.62x10^2$	0.939
Halton FS	300	$538.00x10^2$	0.936
Equidistant FS	0	$6.10x10^2$	0

Table C.11: Final results obtained for each approach in Hydrogen case

500 seconds	# Solutions	Hypervolume
Sobol RS	18	0.949
Random RS	22	0.942
Halton RS	15	0.941
Equidistant RS	24	0.939
Sobol FS	3	0.656
Halton FS	2	0.557
Random FS	2	0.396
Equidistant FS	0	0

Table C.12: Intermediate results computed up to 500 seconds in Hydrogen case

C.2 Figures

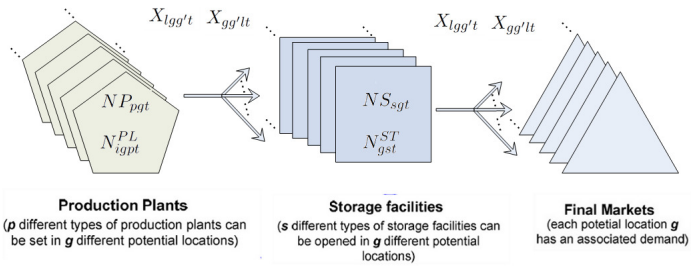


Figure C.1: Decision variables used by the LNS algorithm.

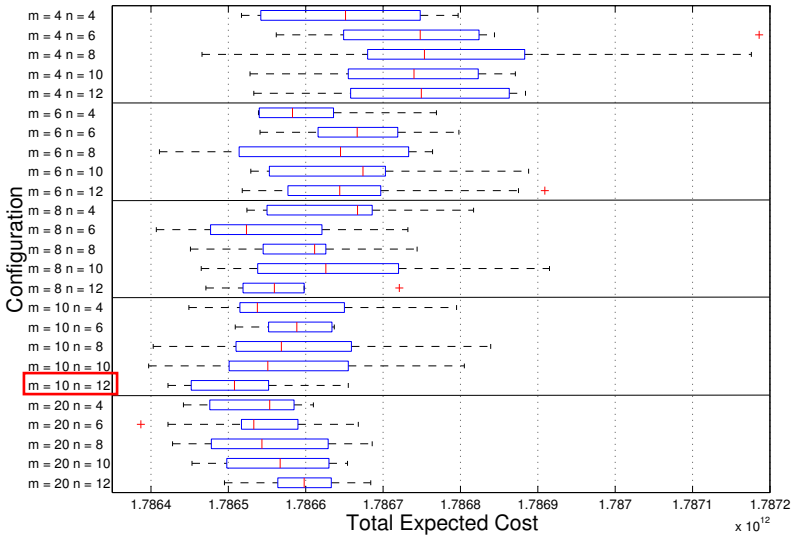


Figure C.2: Tuning of hydrogen results sorted by m and n

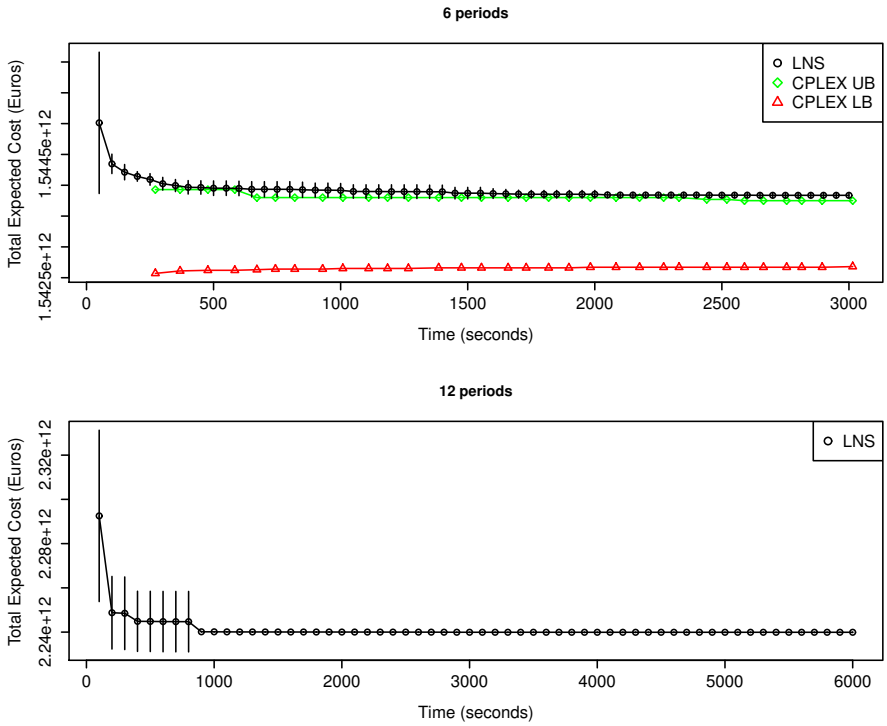


Figure C.3: LNS compared with CPLEX for model hydrogen for $t = 6$ and $t = 12$. The vertical bars show the standard deviation of LNS over 10 runs. For $t > 6$ CPLEX was not able to obtain a solution.

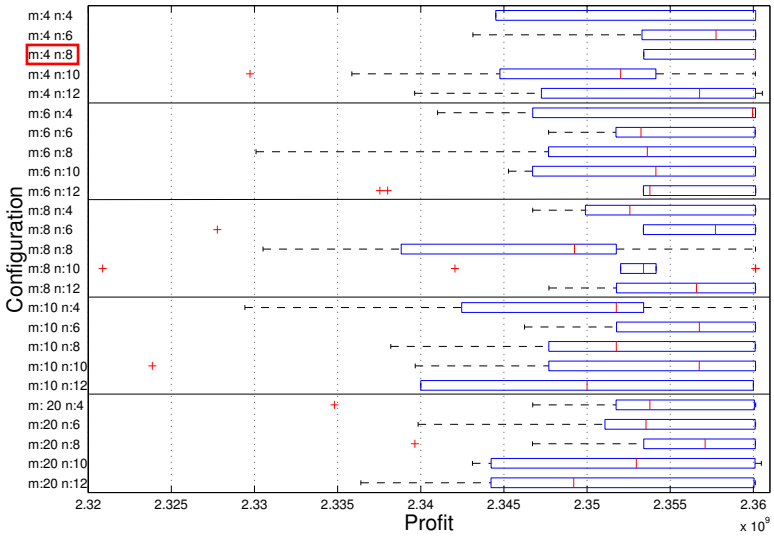


Figure C.4: Tuning of ethanol results sorted by m and n

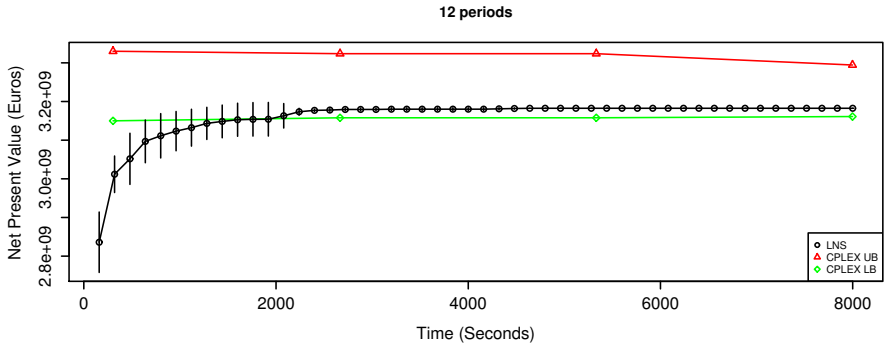


Figure C.5: LNS compared with CPLEX for model ethanol for $t = 12$. The vertical bars show the standard deviation of LNS over 10 runs.

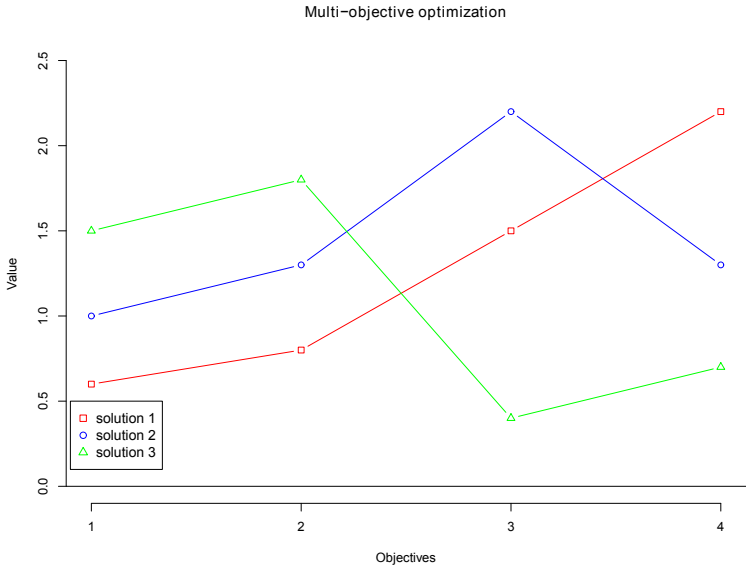


Figure C.6: Illustrative example of dominance structure.

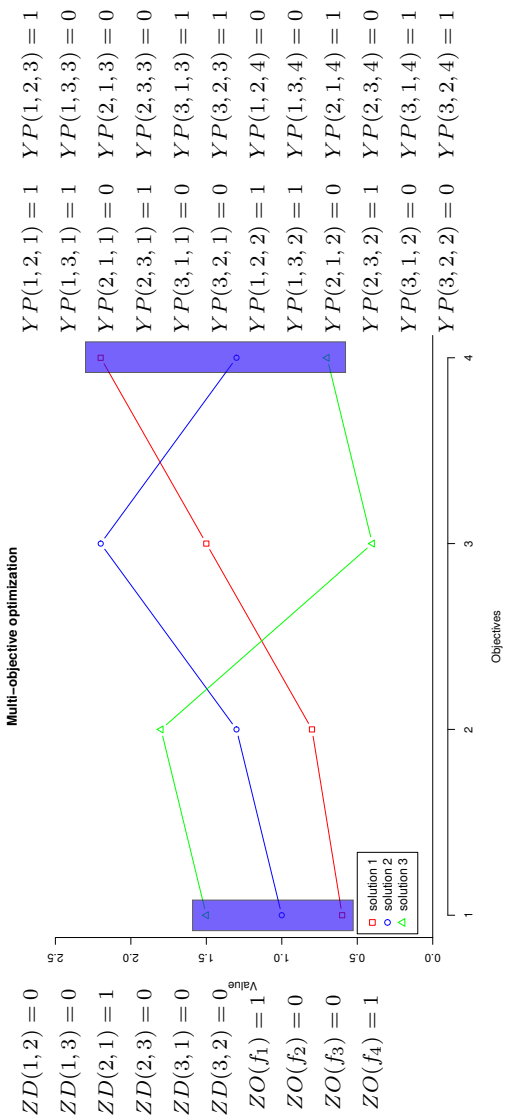


Figure C.7: Example of the notation used.

[ht]

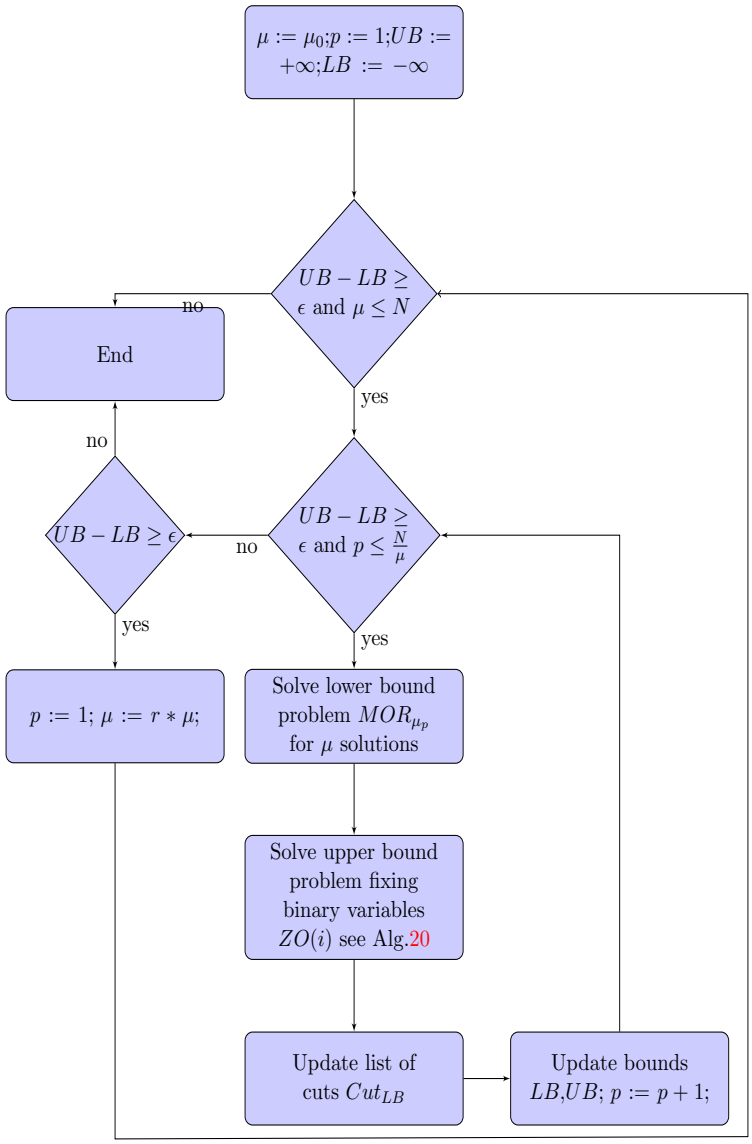
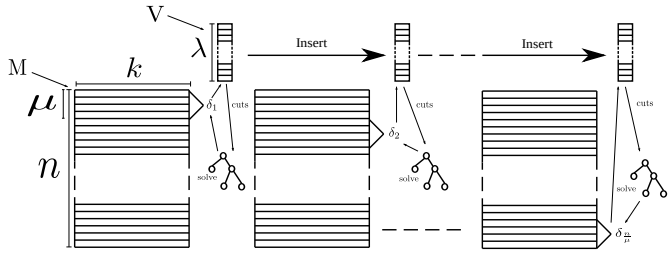
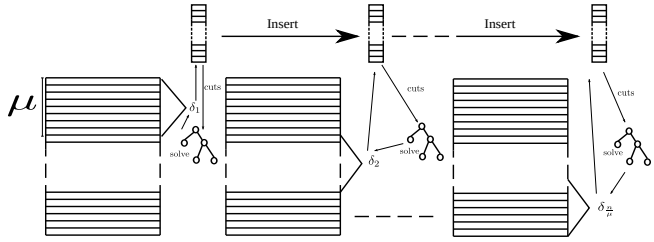


Figure C.8: Flowchart of the decomposition algorithm.

Iteration 1: $\mu = 4$



Iteration 2: $\mu = 8$



Iteration $lg_2(n)$: $\mu = n$

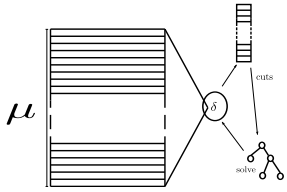


Figure C.9: Illustrative example of how the algorithm works. In each iteration we solve a problem of size μ. The solution of this problem is located in the first position in the list $Cuts_{LB}$. The $Cuts_{LB}$ list is sorted in a descendent order of delta values. Finally, the cuts are generated and added to the model.

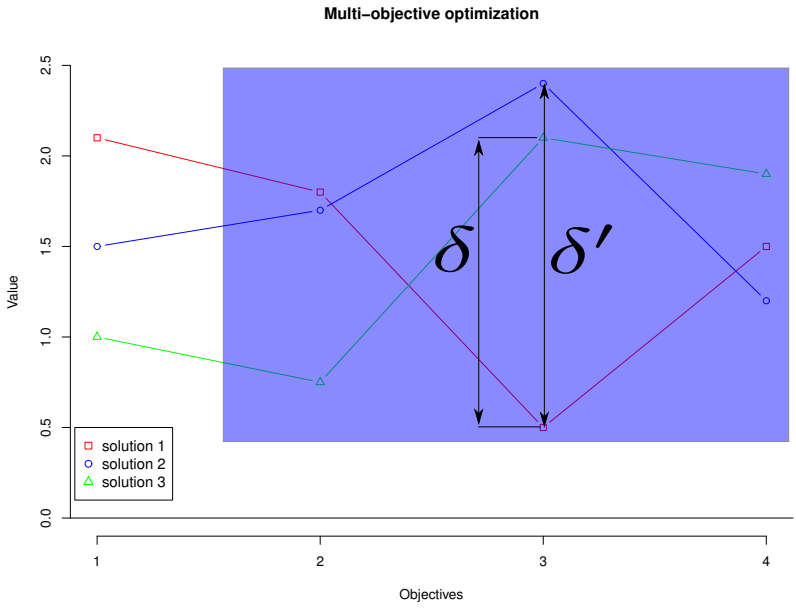


Figure C.10: Illustrative example on how to define the delta error (according to either Brockhoff and Zitzler (Brockhoff and Zitzler, 2006a), or to Guillén-Gosálbez, (Guillén-Gosálbez, 2011a). In this example, we remove objectives two, three and four. According to Brockhoff and Zitzler (Brockhoff and Zitzler, 2006a), the error would be δ', as we consider the error between any two solutions such that one dominates the other in the reduced space. However, according to Guillén-Gosálbez (Guillén-Gosálbez, 2011a), the error would be δ.

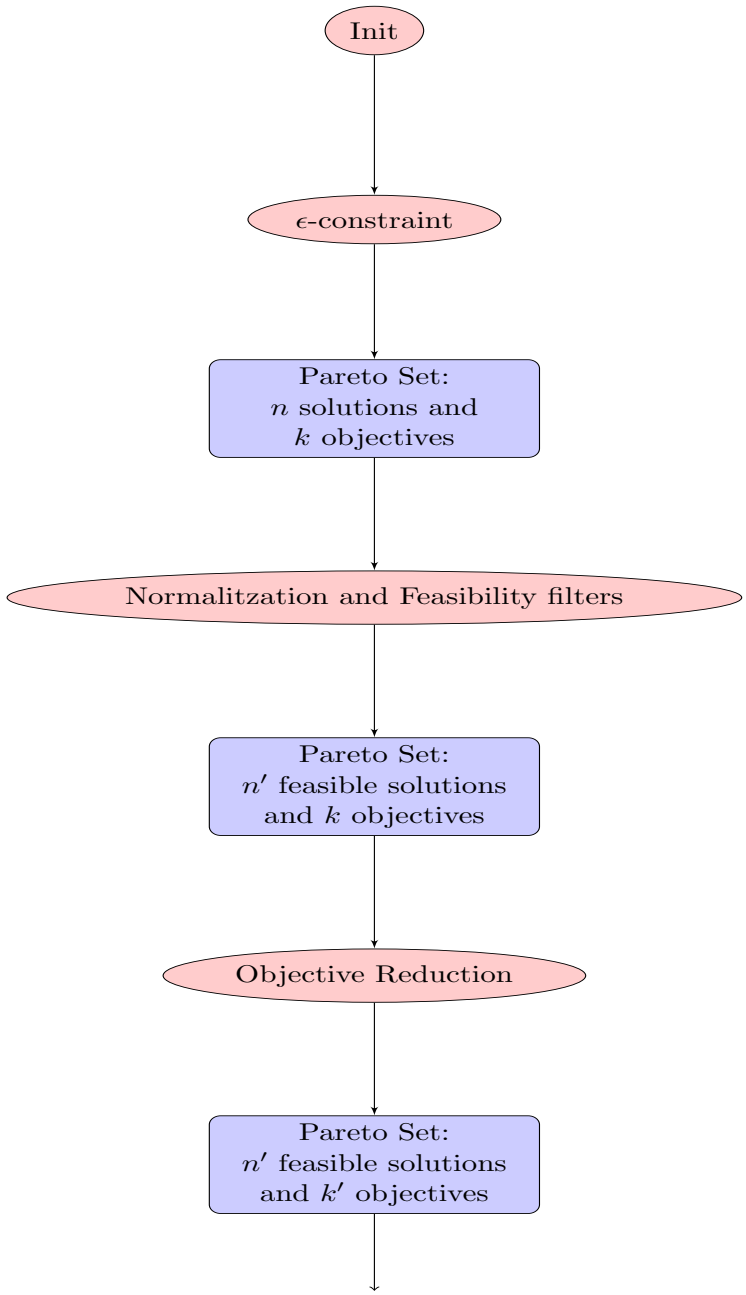


Figure C.11: Proposed framework to Pareto set generation.

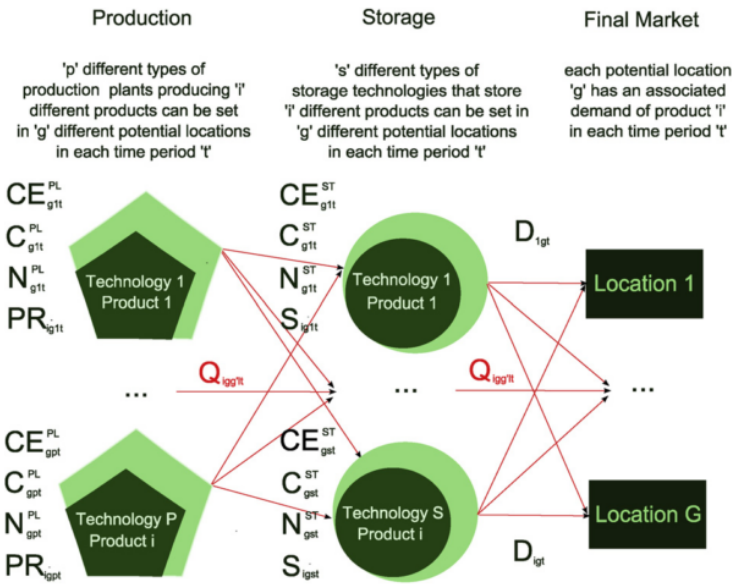


Figure C.12: Superstructure for the hydrogen supply chain design problem, which is formulated as an MILP. We derive an MILP for the optimal design of the network that optimizes simultaneously the total cost and environmental performance (quantified in terms of 16 environmental impacts). The MILP provides the optimal location of production and storage facilities, the technologies to be implemented and the transportation links between the SC entities.

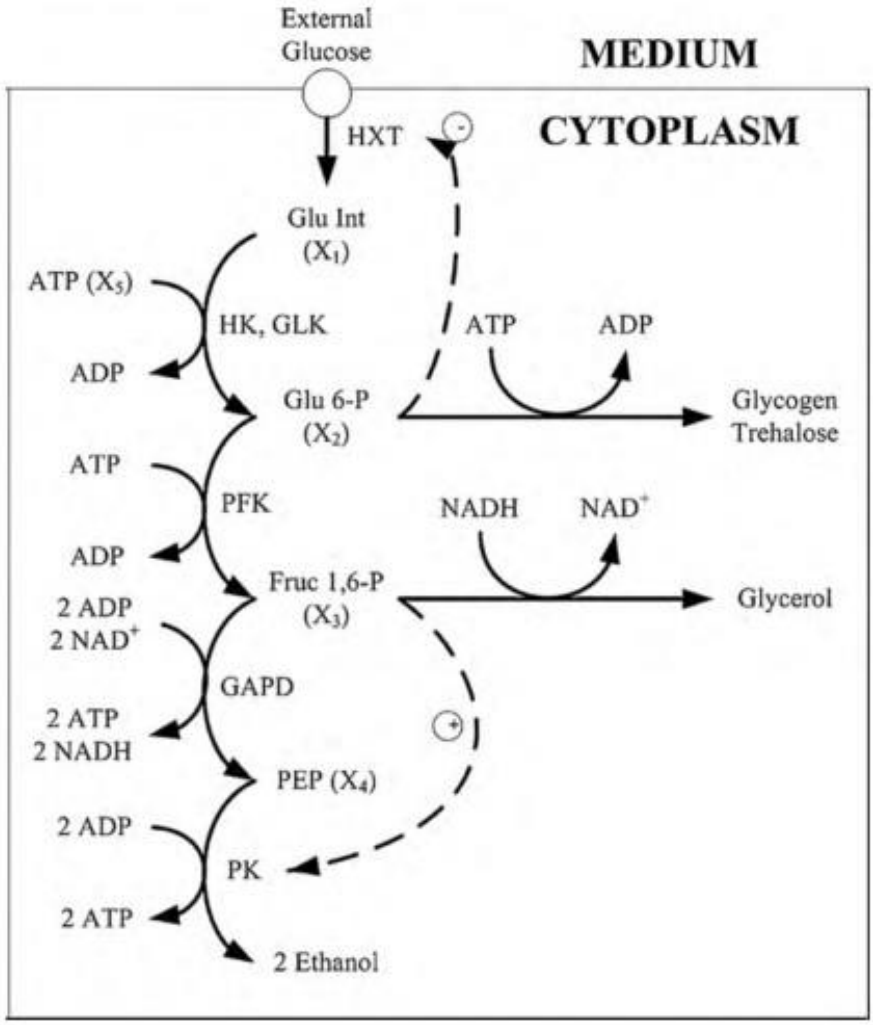


Figure C.13: Metabolic pathway of the fermentation of Saccharomyces cerevisiae. The problem of identifying optimal enzymatic profile in terms of several biological criteria is posed as an MINLP based on a generic kinetic representation of the network. The objectives considered include the minimization of the enzymatic manipulations, the maximization of the ethanol synthesis rate and the minimization of several intermediate metabolites.

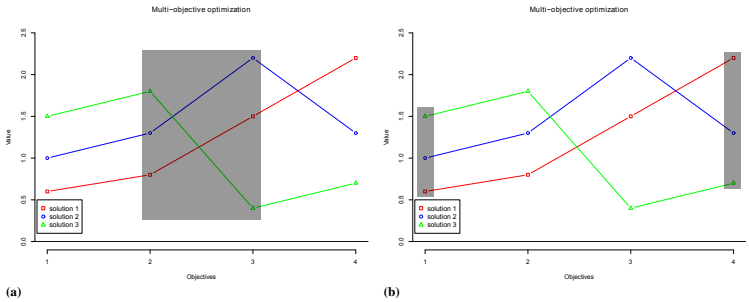


Figure C.14: Illustrative example of objective redaction, where we consider the 3 Pareto optimal solutions that minimize 4 different objectives: $(1, \ldots, 4)$. As observed in (a), the objectives 2 and 3 can be omitted (gray rectangles) without changing the dominance structure. This is because $x \preceq_{f_1,f_4} y$ is satisfied if and only if $x \preceq_{f_1,f_2,f_3,f_4} y$ is satisfied. But as can be observed in (b), further reductions are not possible without modifying the dominance structure.

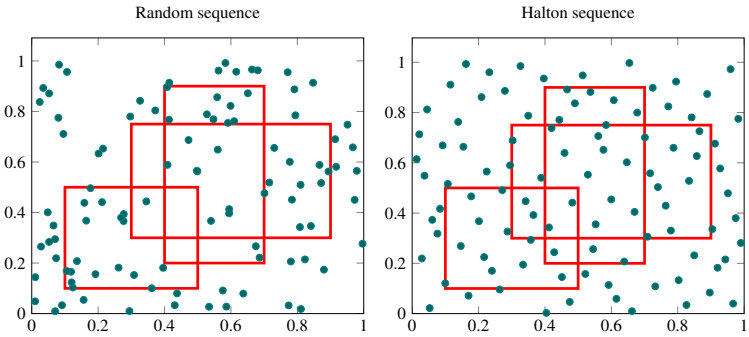


Figure C.15: Illustrative example of how to define discrepancy. The area of union rectangles is known, which is 0.44, but it can approximated by means of counting those points amongst 100 that fall into the rectangles, whether these points are distributed according a pseudo-random sequence, we obtain a discrepancy of $|\frac{39}{100} - 0.44| = 0.05$, whereas whether these points are distributed according to Halton sequence , the discrepancy computed is $|\frac{44}{100} - 0.44| = 0$.

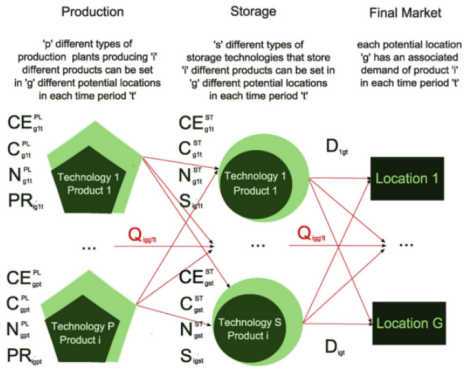


Figure C.16: Superstructure for the supply chain design problem, which is formulated as an MILP.We derive an MILP for the optimal design of the network that optimizes simultaneously the net present value/total cost and environmental performance. The MILP provides the optimal location of the production and storage facilities, the technologies to be implemented and the transportation links between the SC entities.

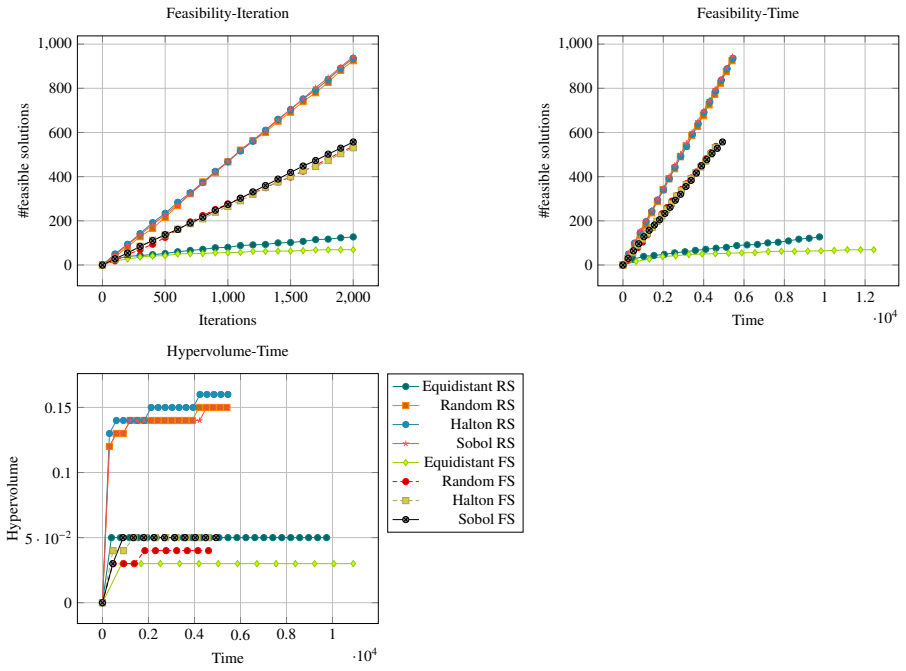


Figure C.17: Feasibility analysis and hypervolume for the case of sustainable planning of ethanol supply chains in each sampling techniques for 6 objectives and 4 objectives: (a) Number of unique feasible olutions vs number of iterations. (b) Number of unique feasible solutions vs time. (c) Hypervolume vs CPU time.

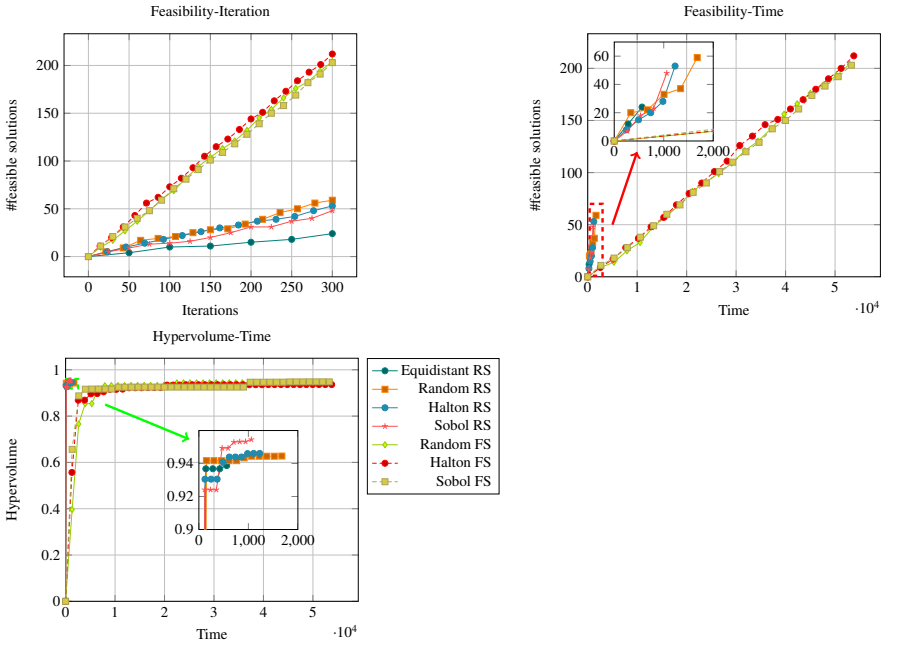


Figure C.18: Feasibility analysis and hypervolume for the case of optimal design of a hydrogen supply chains in each sampling techniques for 16 objectives and 3 objectives: (a) Number of unique feasible solutions vs number of iterations. (b) Number of unique feasible solutions vs time. (c) Hypervolume vs CPU time. ϵ-constraint was not able to obtain feasible solutions with equidistant technique in full space.

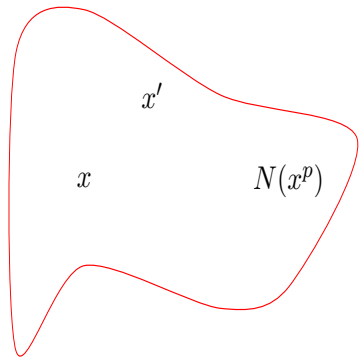


Figure C.19: The neighbourhood search space $N(x^p)$ from solution x, where x' is the optimal solution of $N(x^P)$, and x^p is the partial solution devided from x.

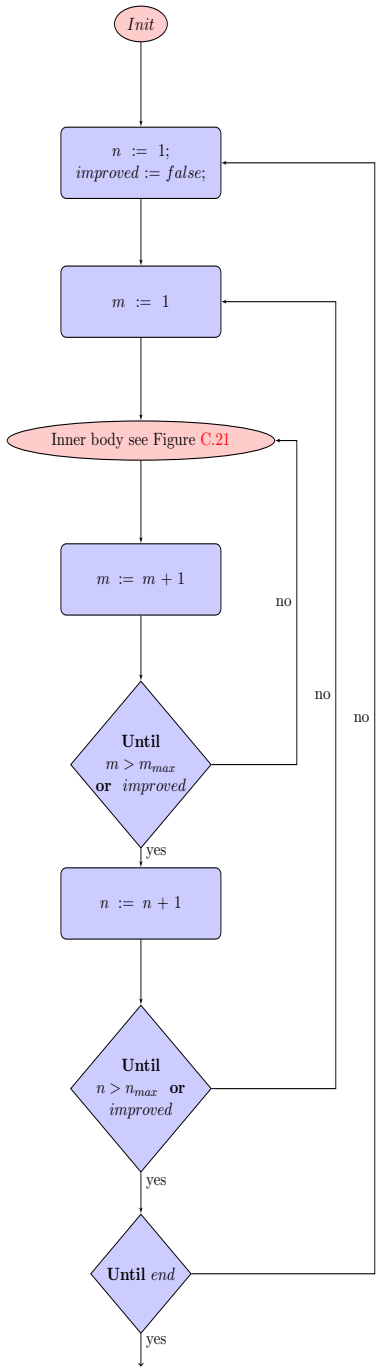


Figure C.20: Flowchart of our tailored LNS for SC.

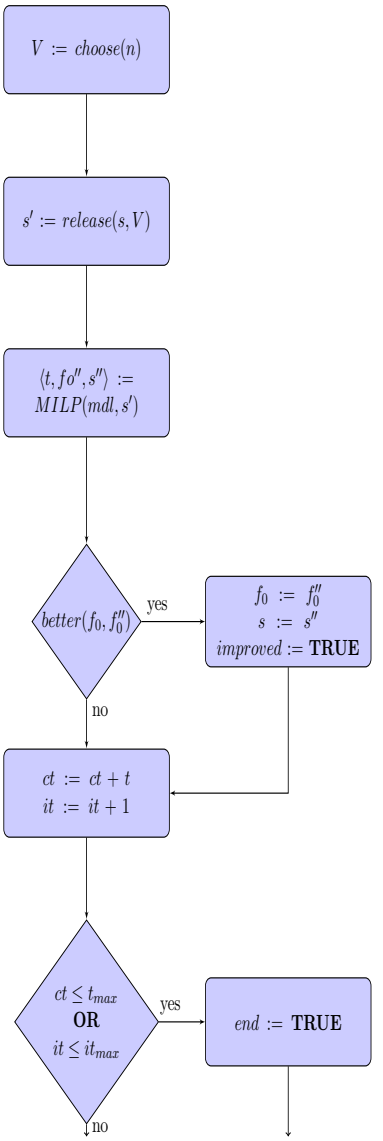


Figure C.21: Inner body of LNS algorithm.

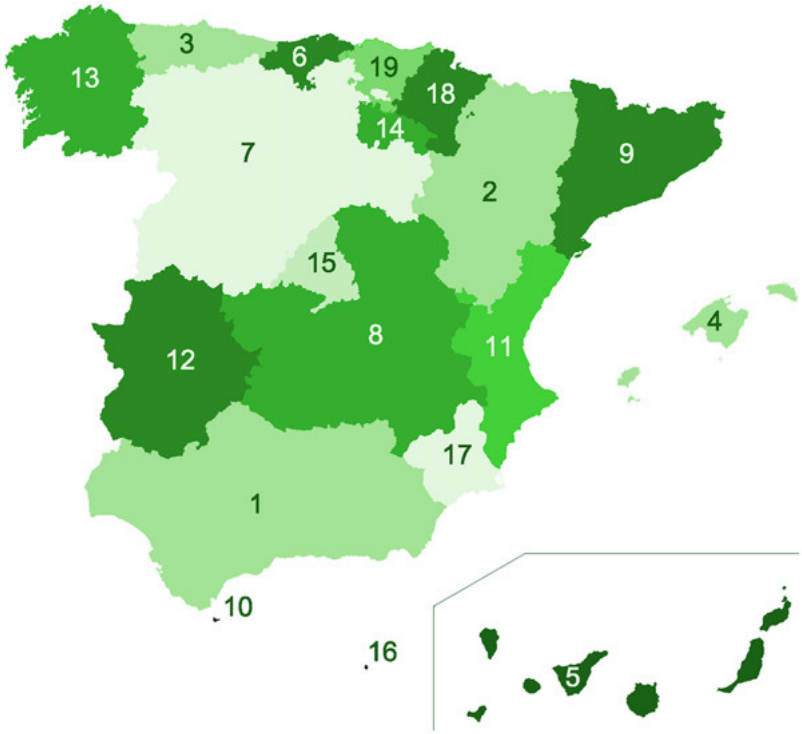


Figure C.22: Autonomous communities of Spain.

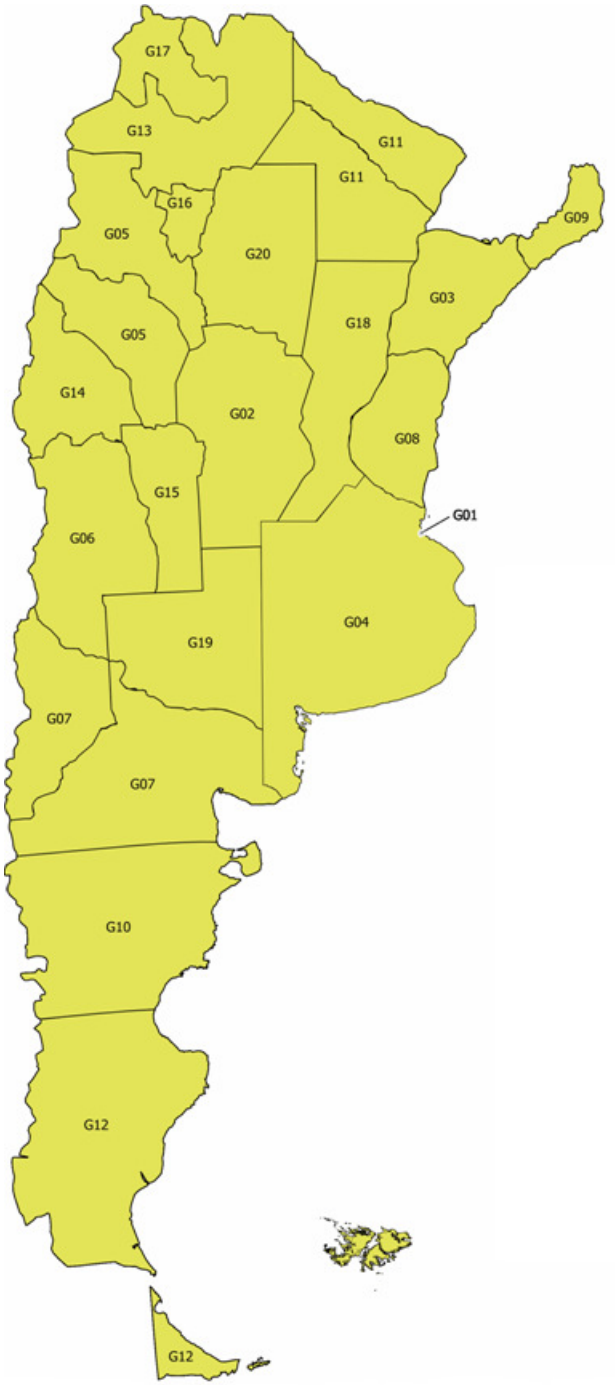


Figure C.23: Provinces of Argentina.

Bibliography

Ahuja, R. K., Ergun, O., Orlin, J. B., Punnen, A. P., 2002. A survey of very large-scale neighborhood search techniques. Discrete Applied Mathematics 123 (1-3), 75–102.

Akgul, O., Zamboni, A., Bezzo, F., Shah, N., Papageorgiou, L. G., 2011. Optimization-based approaches for bioethanol supply chains. Industrial and Engineering Chemistry Research 50 (9), 4927–4938.

Alba, E. (Ed.), 2005. Parallel Metaheuristics: A New Class of Algorithms. John Wiley.

Almansoori, A., Shah, N., 2006. Design and operation of a future hydrogen supply chain: Snapshot model. Chemical Engineering Research and Design 84 (6 A), 423–438.

Almansoori, A., Shah, N., 2009. Design and operation of a future hydrogen supply chain: Multi-period model. International Journal of Hydrogen Energy 34 (19), 7883–7897.

Alonso M., A. H. A.-O. C., 2012. A multiobjective approach for reactive power planning in networks with wind power generation. Renewable Energy 37 (1), 180–191.

Amodeo, L., Prins, C., Sánchez, D. R., 2009. Comparison of metaheuristic approaches for multi-objective simulation-based optimization in supply chain inventory management. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5484 LNCS, 798–807.

Applegate, D., Cook, W., 1991. A computational study of the job-shop scheduling problem. ORSA Journal on Computing 3 (2), 149–156.

Azapagic, a., Clift, R., Dec. 1999. The application of life cycle assessment to process optimisation. Computers & Chemical Engineering 23 (10), 1509–1526.

Balasubramanian, J., Grossmann, I. E., 2004. Approximation to multistage stochastic optimization in multiperiod batch plant scheduling under demand uncertainty. Industrial and Engineering Chemistry Research 43 (14), 3695–3713.

Balat, M., Kirtay, E., 2010. Major technical barriers to a "hydrogen economy". Energy Sources, Part A: Recovery, Utilization and Environmental Effects 32 (9), 863–876.

Banga, J., 2008. Optimization in computational systems biology. BMC Systems Biology 2 (1), 47.

Baykasoglu, A., Gocken, T., 2010. Multi-objective aggregate production planning with fuzzy parameters. Advances in Engineering Software 41 (9), 1124–1131.

Bazaraa, M. S., Sherali, H. D., Shetty, C. M., 2006. Nonlinear programming: theory and algorithms. John Wiley and Sons.

Beume, N., 2006. Faster S-Metric Calculation by Considering Dominated Hypervolume as Klee ' s Measure Problem.

Blum, C., 2004. Theoretical and practical aspects of ant colony optimization. IOS Press.

Blum, C., 2005. Ant colony optimization: Introduction and recent trends. Physics of Life reviews 2 (4), 353–373.

Blum, C., 2005. {Beam-ACO}—{H}ybridizing Ant Colony Optimization with Beam Search: {A}n Application to Open Shop Scheduling. Computers {&} Operations Research 32 (6), 1565–1591.

Blum, C., Aguilera, M. J. B., Roli, A., 2008. Hybrid metaheuristics: an emerging approach to optimization. Vol. 114. Springer Verlag, Warsaw.

Blum, C., Puchinger, J., Raidl, G. R., Roli, A., Sep. 2011. Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing 11 (6), 4135–4151.

Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR) 35 (3), 268–308.

Blum C. Puchinger, J., Raidl, G. R., Roli, A., 2011. Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing Journal 11 (6), 4135–4151.

Bojarski, A. D., Laínez, J. M., Espuña, A., Puigjaner, L., 2009. Incorporating environmental impacts and regulations in a holistic supply chains modeling: An LCA approach. Computers & Chemical Engineering 33 (10), 1747–1759.

Bok, J., Grossmann, I. E., Park, S., 2000. Supply chain optimization in continuous flexible process networks. Industrial and Engineering Chemistry Research 39 (5), 1279–1290.

Bratley, P., Fox, B. L., Mar. 1988. ALGORITHM 659: implementing Sobol's quasirandom sequence generator. ACM Transactions on Mathematical Software 14 (1), 88–100.

Bratley, P., Fox, B. L., Niederreiter, H., Jul. 1992. Implementation and tests of low-discrepancy sequences. ACM Transactions on Modeling and Computer Simulation 2 (3), 195–213.

Brockhoff, D., Zitzler, E., 2006a. Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. Parallel Problem Solving from Nature-PPSN IX, 533–542.

Brockhoff, D., Zitzler, E., 2006b. Dimensionality reduction in multiobjective optimization with (partial) dominance structure preservation: Generalized minimum objective subset problems. TIK Report 247.

Brockhoff, D., Zitzler, E., 2006c. On Objective Conflicts and Objective Reduction in Multiple Criteria Optimization. TIK Report 243.

Buxton, A., Pistikopoulos, E. N., 2004. Environmental impact minimization

through material substitution: a multi-objective optimization approach, 407–417.

C.H. Papadimitriou and K. Steiglitz, 1982. Combinatorial Optimization - Algorithms and complexity.

Charnes, A., Clower, R. W., Kortanek, K. O., 1967. Effective Control Through Coherent Decentralization with Preemptive Goals. Econometrica 35 (2), pp. 294–320.

Charnes, A., Cooper, W. W., 1961. Management models and industrial applications of linear programming [by] A. Charnes [and] WW Cooper. Vol. 1. John Wiley & Sons.

Charnes, A., Cooper, W. W., Ferguson, R. O., Jan. 1955. Optimal Estimation of Executive Compensation by Linear Programming. Management Science 1 (2), 138–151.

Chen, P., Pinto, J. M., 2008. Lagrangean-based techniques for the supply chain management of flexible process networks. Computers and Chemical Engineering 32 (11), 2505–2528.

Chiang, W. ., Russell, R., Xu, X., Zepeda, D., 2009. A simulation/metaheuristic approach to newspaper production and distribution supply chain problems. International Journal of Production Economics 121 (2), 752–767.

Claro J., P. D. S. J., 2012. A multiobjective metaheuristic for a mean-risk multistage capacity investment problem with process flexibility. Computers and Operations Research 39 (4), 838–849.

Cloquell V, Santamarina M, H. A., 2001. Nuevo procedimiento para la normalizacin de valores numricos en la toma de decisiones. In: XVII Congreso Nacional de Ingeniera de Proyectos. Murcia.

Copado-Méndez, P. J., Guillén-Gosálbez, G., Jiménez, L., Apr. 2014. MILP-based decomposition algorithm for dimensionality reduction in multi-objective optimization: Application to environmental and systems biology problems. Computers & Chemical Engineering.

Dal-Mas, M., Giarola, S., Zamboni, A., Bezzo, F., 2011. Strategic design and investment capacity planning of the ethanol supply chain under price uncertainty. Biomass and Bioenergy 35 (5).

Danna, E., Rothberg, E., Pape, C. L., 2005. Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming 102 (1), 71–90.

Dantzig, G. B., 1963. Linear programming and extensions. Princeton University, Princeton, NJ.

Das, I., Dennis, J., 1996. AN ALTERNATE METHOD FOR GENERATING PARETO OPTIMAL POINTS IN MULTICRITERIA OPTIMIZATION (96).

Dave, U., Dantzig, G. B., Thapa, M. N., Nov. 1998. Linear Programming-1: Introduction. The Journal of the Operational Research Society 49 (11), 1226.

de Hijas-Liste, G. M., Klipp, E., Balsa-Canto, E., Banga, J. R., Jan. 2014.

Global dynamic optimization approach to predict activation in metabolic pathways. BMC systems biology 8, 1.

De-León Almaraz, S., Azzaro-Pantel, C., Montastruc, L., Pibouleau, L., Senties, O. B., Nov. 2013. Assessment of mono and multi-objective optimization to design a hydrogen supply chain. International Journal of Hydrogen Energy 38 (33), 14121–14145.

Deb, K., Saxena, D. K., 2005. On finding Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. KanGal Report Number 2005011.

Delavar, M. R., Hajiaghaei-Keshteli, M., Molla-Alizadeh-Zavardehi, S., 2010. Genetic algorithms for coordinated scheduling of production and air transportation. Expert Systems with Applications 37 (12).

Della Croce, F., T'kindt, V., 2002. A {R}ecovering {B}eam {S}earch algorithm for the one machine dynamic total completion time scheduling problem. Journal of the Operational Research Society 53 (11), 1275–1280.

Dimitriadis, A. D., Shah, N., Pantelides, C. C., 1997. {RTN}-based rolling horizon algorithms for medium term scheduling of multipurpose plants. Computers and Chemical Engineering 21 (SUPPL.1), S1061–S1066.

Dogan, K., Goetschalckx, M., 1999. A primal decomposition method for the integrated design of multi-period production-distribution systems. IIE Transactions (Institute of Industrial Engineers) 31 (11), 1027–1036.

Dorigo, M., Blum, C., 2005. Ant colony optimization theory: A survey. Theoretical computer science 344 (2-3), 243–278.

Du, Y., Xie, L., Liu, J., Wang, Y., Xu, Y., Wang, S., Jan. 2014. Multi-objective optimization of reverse osmosis networks by lexicographic optimization and augmented epsilon constraint method. Desalination 333 (1), 66–81.

Dunnett, A. J., Adjiman, C. S., Shah, N., 2008. A spatially explicit whole-system model of the lignocellulosic bioethanol supply chain: An assessment of decentralised processing potential. Biotechnology for Biofuels 1.

Duque, J., Barbosa-Póvoa, A., Novais, A. Q., 2010. Design and planning of sustainable industrial networks: Application to a recovery network of residual products. Industrial & Engineering Chemistry Research 49 (9), 4230–4248.

Duran, M. A., Grossmann, I. E., 1986. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical Programming 36 (3), 307–339.

Ehrgott, M., Ryan, D. M., 2003. Constructing Robust Crew Schedules with Bicriteria Optimization 150 (2002), 139–150.

Elghali, L., Clift, R., Sinclair, P., Panoutsou, C., Bauen, A., 2007. Developing a sustainability framework for the assessment of bioenergy systems. Energy Policy 35 (12), 6075–6083.

Elkamel, A., Mohindra, A., 1999. Rolling horizon heuristic for reactive scheduling of batch process operations. Engineering Optimization 31 (6), 763–792.

Everson, R. M., Fieldsend, J. E., Singh, S., 2002. Full Elite Sets for Multi-objective Optimisation (Acdm), 1–12.

Faure, H., Lemieux, C., Jan. 2010. Improved Halton sequences and discrepancy bounds. Monte Carlo Methods and Applications 16 (3-4), 1–18.

Feo, T. A., Resende, M. G. C., 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization 6 (2), 109–133.

Focacci, F., Laburthe, F., Lodi, A., 2002. Local {S}earch and {C}onstraint {P}rogramming. In: Glover, F., Kochenberger, G. (Eds.), Handbook of Metaheuristics. Vol. 57 of International Series in Operations Research & Management Science. Kluwer Academic Publishers, Norwell, MA.

Fogel, L. J., Owens, A. J., Walsh, M. J., Others, 1966. Artificial intelligence through simulated evolution 26.

Fox, B. L., 1986. ALGORITHM 647: Implementation and Relative Efficiency of Quasirandom Sequence Generators 12 (4), 362–376.

Fox, M. S., Barbuceanu, M., Teigen, R., 2000. Agent-oriented supply-chain management. International Journal of Flexible Manufacturing Systems 12 (2), 165–188.

Frota Neto, J. Q., Bloemhof-Ruwaard, J. M., Van Nunen, J., Van Heck, E., 2008. Designing and evaluating sustainable logistics networks. International Journal of Production Economics 111 (2), 195–208.

Fu, M. C., 2002. Optimization for simulation: Theory vs. practice. INFORMS Journal on Computing 14 (3), 192–215.

Gebreslassie, B. H., Waymire, R., You, F., 2013. Sustainable Design and Synthesis of Algae-Based Biorefinery for Simultaneous Hydrocarbon Biofuel Production and Carbon Sequestration 59 (5), 1599–1621.

Gendreau, M., Potvin, J.-Y., 2010. Handbook of Metaheuristics (2thEd).

Geoffrion, A. M., 1972. Generalized benders decomposition. Journal of optimization theory and applications 10 (4), 237–260.

Geoffrion, A. M., Graves, G. W., 1974. MULTICOMMODITY DISTRIBUTION SYSTEM DESIGN BY BENDERS DECOMPOSITION. Management Science 20 (5), 822–844.

Giarola, S., Shah, N., Bezzo, F., 2012a. A comprehensive approach to the design of ethanol supply chains including carbon trading effects. Bioresource Technology 107, 175–185.

Giarola, S., Zamboni, A., Bezzo, F., 2011. Spatially explicit multi-objective optimisation for design and planning of hybrid first and second generation biorefineries. Computers and Chemical Engineering 35 (9), 1782–1797.

Giarola, S., Zamboni, A., Bezzo, F., 2012b. Environmentally conscious capacity planning and technology selection for bioethanol supply chains. Renewable Energy 43, 61–72.

Glober, F., Kochenberger, G. A., 2003. Handbook of Metaheuristics.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research 13 (5), 533–549.

Grossmann, I., 2005. Enterprise-wide optimization: A new frontier in process systems engineering. AIChE Journal 51 (7), 1846–1857.

Guillén, G., Mele, F. D., Espuña, A., Puigjaner, L., 2006. Addressing the design of chemical supply chains under demand uncertainty. Industrial and Engineering Chemistry Research 45 (22), 7566–7581.

Guillén-Gosálbez, G., 2008. Optimal design and planning of sustainable chemical supply chains under uncertainty.

Guillén-Gosálbez, G., 2011a. A novel MILP-based objective reduction method for multi-objective optimization: application to environmental problems. Computers & Chemical Engineering.

Guillén-Gosálbez, G., Aug. 2011b. A novel MILP-based objective reduction method for multi-objective optimization: Application to environmental problems. Computers & Chemical Engineering 35 (8), 1469–1477.

Guillén-Gosálbez, G., Grossmann, I. E., 2009. Optimal design and planning of sustainable chemical supply chains under uncertainty. AIChE Journal 55 (1), 99–121.

Guillén-Gosálbez, G., Mele, F. D., Grossmann, I. E., 2010. A bi-criterion optimization approach for the design and planning of hydrogen supply chains for vehicle use. AIChE Journal 56 (3), 650–667.

Gupta, A., Maranas, C. D., 1999. A hierarchical Lagrangean relaxation procedure for solving midterm planning problems. Industrial and Engineering Chemistry Research 38 (5), 1937–1947.

Gupta, O. K., Ravindran, A., 1985. Branch and bound experiments in convex nonlinear integer programming. Management Science 31 (12), 1533–1546.

Haimes, Y.Y.; Lasdon, L.S.; Wismer, D., 1971. Integrated Optimization. IEE Transactions on system, man, and cybernetics 47 (JULy), 296–297.

Halton, J. H., 1960. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numer. Mathem. 2, 84–90.

Hansen, P., Mladenovi, N., 2001. Variable neighborhood search: Principles and applications. European journal of operational research 130 (3), 449–467.

Higuera, C., Villaverde, A. F., Banga, J. R., Ross, J., Morán, F., Jan. 2012. Multi-criteria optimization of regulation in metabolic networks. PloS one 7 (7), e41122.

Hischier, R., Bauer, C., Doka, G., Dones, R., Frischknecht, R., Hellweg, S., Jungbluth, N., Loerincik, Y., Margni, M., Nemecek, T., 2010. Implementation of Life Cycle Impact Assessment Methods (3).

Holland, J. H., 1992. Adaptation in natural and artificial systems.

Hugo, a., Pistikopoulos, E., Dec. 2005. Environmentally conscious long-range planning and design of supply chain networks. Journal of Cleaner Production 13 (15), 1471–1491.

Ibaraki, T., Nakamura, K., 2006. Packing Problems with Soft Rectangles. In: Almeida, F., Blesa, M., Blum, C., Moreno, J. M., Pérez, M., Roli, A.,

Sampels, M. (Eds.), Proceedings of HM 2006 – 3rd International Workshop on Hybrid Metaheuristics. Vol. 4030 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, pp. 13–27.

IBM ILOG, C., 2012. CPLEX 12.

Ijiri, Y., 1965. Management goals and accounting for control. Studies in mathematical and managerial economics. Amsterdam: North Holland.

inc. Lindo Systems, 2012. LINGO user's guide.

INGASON, H., PALLINGOLFSSON, H., JENSSON, P., Jul. 2008. Optimizing site selection for hydrogen production in Iceland. International Journal of Hydrogen Energy 33 (14), 3632–3643.

Jain, A., Palekar, U. S., 2005. Aggregate production planning for a continuous reconfigurable manufacturing process. Computers and Operations Research 32 (5), 1213–1236.

Kim, J., Lee, Y., Moon, I., 2008. Optimization of a hydrogen supply chain under demand uncertainty. International Journal of Hydrogen Energy 33 (18), 4715–4729.

Klee, V., 1977. Can the Measure of ∪n 1 [ai, bi] be Computed in Less Than O(n log n) Steps? The American Mathematical Monthly 84 (4), pp. 284–285.

Koroneos, C., Dompros, A., Roumbas, G., Moussiopoulos, N., 2004. Life cycle assessment of hydrogen fuel production processes. International Journal of Hydrogen Energy 29 (14), 1443–1450.

Kostin, A., Guille, G., Mele, F. D., Jime, L., 2012. Identifying Key Life Cycle Assessment Metrics in the Multiobjective Design of Bioethanol Supply Chains Using a Rigorous Mixed-Integer Linear Programming Approach.

Kostin, A. M., Guillén-Gosálbez, G., Mele, F. D., Bagajewicz, M. J., Jiménez, L., 2010. Integrating pricing policies in the strategic planning of supply chains: A case study of the sugar cane industry in Argentina. Computer Aided Chemical Engineering 28 (C), 103–108.

Kostin, A. M., Guillén-gosálbez, G., Mele, F. D., Bagajewicz, M. J., Jiménez, L., 2011a. A novel rolling horizon strategy for the strategic planning of supply chains . Application to the sugar cane industry of Argentina 35, 2540–2563.

Kostin, A. M., Guillén-Gosálbez, G., Mele, F. D., Bagajewicz, M. J., Jiménez, L., 2011b. A novel rolling horizon strategy for the strategic planning of supply chains. {A}pplication to the sugar cane industry of Argentina. Computers and Chemical Engineering 35 (11), 2540–2563.

Kravanja, Z., Čuček, L., Jan. 2013. Multi-objective optimisation for generating sustainable solutions considering total effects on the environment. Applied Energy 101, 67–80.

Lawler, E. L., Lenstra, J. K., Kan, A., Shmoys, D. B., 1985. The traveling salesman problem: a guided tour of combinatorial optimization. Vol. 3. Wiley New York.

LI, Z., GAO, D., CHANG, L., LIU, P., PISTIKOPOULOS, E., Oct. 2008. Hydrogen infrastructure design and optimization: A case study of China. International Journal of Hydrogen Energy 33 (20), 5275–5286.

Lin, Z., Ogden, J., Fan, Y., Chen, C. W., 2008. The fuel-travel-back approach to hydrogen station siting. International journal of hydrogen energy 33 (12), 3096–3101.

Liu, P.-K., Wang, F.-S., Apr. 2008. Inference of biochemical network models in S-system using multiobjective optimization approach. Bioinformatics (Oxford, England) 24 (8), 1085–92.

Lodi, A., 2010. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: 7th International Conference, CPAIOR 2010, Bologna, Italy, June 14-18, 2010, Proceedings. Vol. 6140. Springer-Verlag New York Inc.

López Jaimes, A., Coello, C., Urías Barrientos, J., 2009. Online objective reduction to deal with many-objective problems. Evolutionary Multi-Criterion Optimization, 423–437.

López Jaimes, A., Coello Coello, C. A., Chakraborty, D., 2008. Objective reduction using a feature selection technique. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation. ACM, pp. 673–680.

Marler, R., Arora, J., Apr. 2004. Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization 26 (6), 369–395.

Mele, F. D., Kostin, A. M., Guillén-Gosálbez, G., Jiménez, L., 2011. Multi-objective model for more sustainable fuel supply chains. A case study of the sugar cane industry in argentina. Industrial and Engineering Chemistry Research 50 (9), 4939–4958.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E., Others, 1953. Equation of state calculations by fast computing machines. The journal of chemical physics 21 (6), 1087.

Mirhassani, S. A., Lucas, C., Mitra, G., Messina, E., Poojari, C. A., 2000. Computational solution of capacity planning models under uncertainty. Parallel Computing 26 (5), 511–538.

Mitrovic-Minic, S., Punnen, A., 2009. Variable Intensity Local Search. Matheuristics: Hybridizing Metaheuristics and Mathematical Programming 10, 245.

Moscato, P., 1999. Memetic algorithms: a short introduction. In: Corne, D., Dorigo, M., Glover, F., Dasgupta, D., Moscato, P., Poli, R., Price, K. V. (Eds.), New ideas in optimization. McGraw-Hill Ltd., UK, Maidenhead, UK, England, pp. 219–234.

Mula, J., Peidro, D., Díaz-Madroñero, M., Vicens, E., 2010. Mathematical programming models for supply chain production and transport planning. European Journal of Operational Research 204 (3), 377–390.

Murillo-alvarado, P. E., El-halwagi, M. M., 2013. Optimization of Pathways for Biore fi neries Involving the Selection of Feedstocks, Products, and Processing Steps.

Murtagh, B. A., Saunders, M. A., 1978. Large-scale linearly constrained optimization. Mathematical Programming 14 (1), 41–72.

Murthy Konda, N., Shah, N., Brandon, N. P., Apr. 2011. Optimal transition towards a large-scale hydrogen infrastructure for the transport sector:

The case for the Netherlands. International Journal of Hydrogen Energy 36 (8), 4619–4635.

Naraharisetti, P. K., Adhitya, A., Karimi, I. A., Srinivasan, R., 2009. From PSE to PSE^{2}-Decision support for resilient enterprises. Computers and Chemical Engineering 33 (12), 1939–1949.

Nemhauser, G. L., Wolsey, L. A., 1988. Integer and combinatorial optimization 18.

Niknam T., Fard A. K., S. A., 2012. Distribution feeder reconfiguration considering fuel cell/wind/photovoltaic power plants. Renewable Energy 37 (1), 213–225.

Ow, P. S., Morton, T. E., 1988. Filtered Beam Search in Scheduling. International Journal of Production Research 26, 297–307.

Papageorgiou, L. G., 2009. Supply chain optimisation for the process industries: Advances and opportunities. Computers and Chemical Engineering 33 (12), 1931–1938.

Paquet, M., Martel, A., Desaulniers, G., 2004. Including technology selection decisions in manufacturing network design models. International Journal of Computer Integrated Manufacturing 17 (2), 117–125.

Pinto-Varela T. Barbosa-Póvoa, A., Novais, A., 2011. Bi-objective optimization approach to the design and planning of supply chains: Economic versus environmental performances. Computers and Chemical Engineering 35 (8), 1454–1468.

Powell, M., 1978. A fast algorithm for nonlinearly constrained optimization calculations. Numerical analysis, 144–157.

Pozo, C., Guillén, G., 2012. Identifying the Preferred Subset of Enzymatic Profiles in Nonlinear Kinetic Metabolic Models via Multiobjective Global Optimization and Pareto Filters. PLOS one 7 (9), 1–11.

Prandtstetter, M., Raidl, G. R., 2008. An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. European Journal of Operational Research 191 (3), 1004–1022.

Puigjaner, L., Espuña, A., Sep. 2006. Computer Aided Process and Product Engineering. Vol. 2. Wiley-VCH Verlag GmbH, Weinheim, Germany.

Puigjaner, L., Guillén-Gosálbez, G., 2008. Towards an integrated framework for supply chain management in the batch chemical process industry. Computers & Chemical Engineering 32 (4-5), 650–670.

Purshouse, R., Fleming, P., 2003. Conflict, harmony, and independence: Relationships in evolutionary multi-criterion optimisation. Evolutionary Multi-Criterion Optimization, 67.

QADRDAN, M., SABOOHI, Y., SHAYEGAN, J., Dec. 2008. A model for investigation of optimal hydrogen pathway, and evaluation of environmental impacts of hydrogen supply system. International Journal of Hydrogen Energy 33 (24), 7314–7325.

Raidl, G. R., 2006. A unified view on hybrid metaheuristics. In: Almeida, F., Blesa, M., Blum, C., Moreno, J. M., Pérez, M., Roli, A., Sampels, M. (Eds.), Proceedings of HM 2006 – 3rd International Workshop on

Hybrid Metaheuristics. Vol. 4030 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, pp. 1–12.

Rechenberg, I., Eigen, M., 1973. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution.

Reeves, C. R., 1993. Modern heuristic techniques for combinatorial problems. John Wiley & Sons, Inc. New York, NY, USA.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40 (4), 455–472.

Sabio, N., Gadalla, M., Guillén-Gosálbez, G., Jiménez, L., 2010. Strategic planning with risk control of hydrogen supply chains for vehicle use under uncertainty in operating costs: A case study of Spain. International Journal of Hydrogen Energy 35 (13), 6836–6852.

Sabio, N., Gadalla, M., Jiménez L.nez, L., Guillén-Gosálbez, G., 2009. Risk management on the design and planning of a hydrogen supply chain for vehicle use under uncertainty in production prices: A case study of Spain. Conference Proceedings - 2009 AIChE Annual Meeting, 09AIChE.

Sabio, N., Kostin, A., Guillén-Gosálbez, G., Jiménez, L., Mar. 2012. Holistic minimization of the life cycle environmental impact of hydrogen infrastructures using multi-objective optimization and principal component analysis. International Journal of Hydrogen Energy 37 (6), 5385–5405.

Sadjadi, S. J., Jafari, M., Amini, T., 2009. A new mathematical modeling and a genetic algorithm search for milk run problem (an auto industry

supply chain case study). International Journal of Advanced Manufacturing Technology 44 (1-2), 194–200.

Santibañez Aguilar, J. E., González-Campos, J. B., Ponce-Ortega, J. M., Serna-González, M., El-Halwagi, M. M., Feb. 2014. Optimal planning and site selection for distributed multiproduct biorefineries involving economic, environmental and social objectives. Journal of Cleaner Production 65, 270–294.

Shapiro, J. F., 2001. Modeling the supply chain. Duxbury.

Sharma A., Y. V., 2012. Modelling and optimization of cut quality during pulsed Nd:YAG laser cutting of thin Al-alloy sheet for straight profile. Optics and Laser Technology 44 (1), 159–168.

Shaw, P., 1998. Using {C}onstraint {P}rogramming and {L}ocal {S}earch {M}ethods to {Solve Vehicle Routing Problems}. In: Maher, M., Puget, J.-F. (Eds.), Principle and Practice of Constraint Programming – CP98. Vol. 1520 of Lecture Notes in Computer Science. Springer, pp. 417–431.

Shmoys, D. B., Lenstra, J. K., Kan, A., Lawler, E. L., 1985. The Traveling Salesman Problem. Wiley Interscience Series in Discrete Mathematics. John Wiley & Sons.

Sobol, I., Jan. 1967. On the distribution of points in a cube and the approximate evaluation of integrals. USSR Computational Mathematics and Mathematical Physics 7 (4), 86–112.

Soner Kara, S., Onut, S., 2010. A stochastic optimization approach for pa-

per recycling reverse logistics network design under uncertainty. International Journal of Environmental Science and Technology 7 (4), 717–730.

Spath, M., Mann, P., 2001. Life cycle assessment of renewable hydrogen production via wind/electrolysis. Tech. rep., Department of Energy's Hydrogen Program.

Spath, P. Mann, M., 2001. Life cycle assessment of hydrogen production via natural gas steam reforming. Tech. rep., National Renewable Energy Laboratory.

Stubbs, R. A., Mehrotra, S., 1999. A branch-and-cut method for 0-1 mixed convex programming. Mathematical Programming 86 (3), 515–532.

Stützle, T., 1999. Local Search Algorithms for Combinatorial Problems - Analysis, Algorithms and New Applications. Ph.D. thesis, TU Darmstadt.

Talbi, E.-G., 2002. A {T}axonomy of {H}ybrid {M}etaheuristics. Journal of Heuristics 8 (5), 541–564.

Thoai, N. V., 2011. Criteria and dimension reduction of linear multiple criteria optimization problems. Journal of Global Optimization, 1–10.

Üster, H., Easwaran, G., Akçali, E., Çetinkaya, S., 2007. Benders decomposition with alternative multiple cuts for a multi-product closed-loop supply chain network design model. Naval Research Logistics 54 (8), 890–907.

Ustun O., K. R., 2012. Combined forecasts in portfolio optimization: A generalized approach. Computers and Operations Research 39 (4), 805–819.

van Hoeve, W. J., Hooker, J. N., 2009. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: 6th International Conference, CPAIOR 2009 Pittsburgh, PA, USA, May 27-31, 2009 Proceedings. Vol. 5547. Springer-Verlag New York Inc.

Vera, J., de Atauri, P., Cascante, M., Torres, N. V., Aug. 2003. Multicriteria optimization of biochemical systems by linear programming: application to production of ethanol by Saccharomyces cerevisiae. Biotechnology and bioengineering 83 (3), 335–43.

Voss, S., Osman, I. H., Roucairol, C., 1999. Meta-heuristics: Advances and trends in local search paradigms for optimization. Kluwer Academic Publishers Norwell, MA, USA.

Voudouris, C., Tsang, E., 1999. Guided local search and its application to the traveling salesman problem. European Journal of Operational Research 113 (2), 469–499.

Wang, F. S., Wu, W. H., May 2013. Multi-Objective Optimization in Chemical Engineering. John Wiley & Sons Ltd, Oxford, UK.

Warren Liao, T., Chang, P. C., 2010. Impacts of forecast, inventory policy, and lead time on supply chain inventoryA numerical study. International Journal of Production Economics 128 (2), 527–537.

Wu, W.-H., Wang, F.-S., Chang, M.-S., Jan. 2011. Multi-objective optimization of enzyme manipulations in metabolic networks considering resilience effects. BMC systems biology 5 (1), 145.

Wu, Y. C., Debs, A. S., Marsten, R. E., 2002. A direct nonlinear predictor-corrector primal-dual interior point algorithm for optimal power flows. Power Systems, IEEE Transactions on 9 (2), 876–883.

You, F., Grossmann, I. E., 2010. Integrated multi-echelon supply chain design with inventories under uncertainty: MINLP models, computational strategies. AICHE Journal 56 (2), 419–440.

Yue, D., Guill, G., You, F., 2013. Global Optimization of Large-Scale Mixed-Integer Linear Fractional Programming Problems: A Reformulation-Linearization Method and Process Scheduling Applications 59 (11), 4255–4272.

Yue, D., Slivinsky, M., Sumpter, J., You, F., Mar. 2014. Sustainable Design and Operation of Cellulosic Bioelectricity Supply Chain Networks with Life Cycle Economic, Environmental, and Social Optimization. Industrial & Engineering Chemistry Research 53 (10), 4008–4029.

Zadeh, L., Jan. 1963. Optimality and non-scalar-valued performance criteria. IEEE Transactions on Automatic Control 8 (1), 59–60.

Zhang, Q., Gong, J., Skwarczek, M., Yue, D., You, F., 2014. Sustainable Process Design and Synthesis of Hydrocarbon Biorefinery through Fast Pyrolysis and Hydroprocessing 60 (3).

Zitzler, E., Thiele, L., 1998. Multiobjective optimization using evolutionary algorithms - A comparative case study. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 1498 LNCS. pp. 292–301.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., da Fonseca, V., Apr. 2003. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation 7 (2), 117–132.