CONSTRAINT AND INTEGER PROGRAMMING
OPERATIONS RESEARCH/COMPUTER SCIENCE
INTERFACES SERIES

Series Editors
Professor Ramesh Sharda
Oklahoma State University

Prof. Dr. Stefan Voß
Universität Hamburg

Other published titles in the series:

Greenberg, Harvey J. / A Computer-Assisted Analysis System for Mathematical Programming
Models and Solutions: A User's Guide for ANALYZE

Greenberg, Harvey J. / Modeling by Object-Driven Linear Elemental Relations: A User's Guide for
MODLER

Brown, Donald/Scherer, William T. / Intelligent Scheduling Systems


Barth, Peter / Logic-Based 0-1 Constraint Programming

Jones, Christopher V. / Visualization and Optimization

Barr, Richard S./ Helgason, Richard V./ Kennington, Jeffery L. / Interfaces in Computer Science & Operations Research: Advances in Metaheuristics, Optimization, & Stochastic Modeling Technologies

Ellacott, Stephen W./ Mason, John C./ Anderson, Iain J. / Mathematics of Neural Networks: Models, Algorithms & Applications

Woodruff, David L. / Advances in Computational & Stochastic Optimization, Logic Programming, and Heuristic Search

Klein, Robert / Scheduling of Resource-Constrained Projects

Bierwirth, Christian / Adaptive Search and the Management of Logistics Systems

Laguna, Manuel / González-Velarde, José Luis / Computing Tools for Modeling, Optimization and Simulation

Stilman, Boris / Linguistic Geometry: From Search to Construction

Sakawa, Masatoshi / Genetic Algorithms and Fuzzy Multiobjective Optimization

Ribeiro, Celso C./ Hansen, Pierre / Essays and Surveys in Metaheuristics

Holsapple, Clyde/ Jacob, Varghese / Rao, H. R. / BUSINESS MODELLING: Multidisciplinary Approaches — Economics, Operational and Information Systems Perspectives

Sleezer, Catherine M./ Wentling, Tim L./ Cude, Roger L. / HUMAN RESOURCE DEVELOPMENT AND INFORMATION TECHNOLOGY: Making Global Connections

Voß, Stefan, Woodruff, David / Optimization Software Class Libraries

Upadhyaya et al / MOBILE COMPUTING: Implementing Pervasive Information and Communications Technologies

Reeves, Colin & Rowe, Jonathan / GENETIC ALGORITHMS—Principles and Perspectives: A Guide to GA Theory

Bhargava, Hemant K. & Ye, Nong / COMPUTATIONAL MODELING AND PROBLEM SOLVING IN THE NETWORKED WORLD: Interfaces in Computer Science & Operations Research

Woodruff, David L./ NETWORK INTERDICTION AND STOCHASTIC INTEGER PROGRAMMING

Anandalingam, G. & Raghavan, S. Raghhu / TELECOMMUNICATIONS NETWORK DESIGN AND MANAGEMENT

Laguna, Manuel & Martí, Rafael / SCATTER SEARCH: Methodology and Implementations in C

Gosavi, Abhijit / SIMULATION-BASED OPTIMIZATION: Parametric Optimization Techniques and Reinforcement Learning

Koutsoukis, Nikitas-Spiros & Mitra, Gautam / DECISION MODELLING AND INFORMATION SYSTEMS: The Information Value Chain
CONSTRAINT AND INTEGER PROGRAMMING
Toward a Unified Methodology

Edited by
MICHELA MILANO
University of Bologna

Springer Science+Business Media, LLC
# Contents

List of Figures \hspace{1cm} xiii

List of Tables \hspace{1cm} xvii

Preface \hspace{1cm} xix

Contributing Authors \hspace{1cm} xxiii

Foreword \hspace{1cm} xxvii

*John Hooker*

\begin{tabular}{ll}
1 & Constraint and Integer Programming \\
& *Michela Milano* and *Michael Trick* \\
1 & Introduction & 1 \\
2 & CP(FD) Basic Concepts & 4 \\
& 2.1 Modeling & 5 \\
& 2.2 Structure of a CP program & 6 \\
& 2.3 Solving & 8 \\
& 2.3.1 Constraint Propagation & 8 \\
& 2.3.2 Search & 10 \\
& 2.3.3 Branch and Bound in CP & 12 \\
& 2.4 An example: the car sequencing problem & 13 \\
3 & Integer Linear Programming Basic Concepts & 15 \\
& 3.1 Modeling & 16 \\
& 3.1.1 Logical Constraints & 16 \\
& 3.1.2 Resource Constraints & 17 \\
& 3.1.3 Routing Constraints & 18 \\
& 3.1.4 Alternative Formulations & 19 \\
& 3.2 Solving & 20 \\
& 3.2.1 Relaxations & 20 \\
& 3.2.2 Branch and Bound & 22 \\
& 3.2.3 Improving the Effectiveness & 23 \\
& 3.3 An example: the car sequencing problem revisited & 26 \\
4 & Incomplete search strategies & 27 \\
5 & Conclusion & 28
\end{tabular}
## Contents

2.1 Preliminaries 94  
2.2 Definition and Advantages 95  
2.3 Examples 96  

3 Filtering Algorithms 104  
3.1 Algorithms Based on Constraints Addition 105  
3.2 General Arc Consistency Filtering Algorithm 106  
3.2.1 Preliminaries 106  
3.2.2 A First Algorithm 107  
3.2.3 A better general algorithm 107  
3.2.4 Discussion and Example 109  
3.3 Dedicated Filtering Algorithms 111  

4 Two Successful Filtering Algorithms 112  
4.1 Preliminaries 113  
4.2 The Alldifferent Constraint 114  
4.2.1 Consistency and Arc Consistency 114  
4.2.2 Complexity 115  
4.2.3 Some Results 116  
4.3 The Global Cardinality Constraint 117  
4.3.1 Consistency and Arc Consistency 117  
4.3.2 Complexity 118  
4.3.3 Some results 118  

5 Global Constraints and Over-constrained Problems 120  
5.1 Satisfiability Sum Constraint 121  
5.2 Global Soft Constraints 122  
5.2.1 General Definitions of Cost 123  
5.2.2 Soft Alldifferent Constraint 124  

6 Quality of Filtering Algorithms 125  
7 Discussion 126  
7.1 Incomplete Algorithms and Fixed-Point Property 126  
7.2 Closure 127  
7.3 Power of a Filtering Algorithm 128  

8 Conclusion 129  

References 131  

5 Exploiting relaxations in CP 137  
Filippo Focacci, Andrea Lodi and Michela Milano  

1 Introduction and Motivation 138  

2 Integer Linear Programming and Relaxations 139  
2.1 Continuous Linear Relaxation 141  
2.2 Structured Relaxations 142  
2.2.1 Surrogate Relaxation 143  
2.2.2 Lagrangean Relaxation 143  

3 Integrating Relaxations in CP 144  
3.1 Which relaxation to use 145  
3.2 Which part of the problem 145  
3.2.1 Relaxation of global constraints 147  

4 Relax to propagate 150
## Contents

4.3 The linear solver interface: *eplex* 184  
4.4 The *repair* solver 185  
4.5 Attributed Variables and Demons in ECLiPSe 185  

5 Programming a Hybrid Search in ECLiPSe 186  
5.1 An Illustrative Example 187  
5.2 Problem Modelling 189  
5.3 Hybrid Probe-based Algorithm 189  
5.3.1 The Algorithm Design Model 191  
5.3.2 Inference phase 191  
5.3.3 Probing phase 192  
5.3.4 Resource feasibility 192  
5.4 Probing Strategies 193  
5.5 Mixed Integer Programming based Probing 194  
5.6 Linear Relaxation based Probing 196  
5.7 Evaluation and Performance Analysis 197  
5.7.1 Setting up the benchmark instances 198  
5.7.2 Computational results 199  

6 Conclusion 200

References 203

7 CP Based Branch-and-Price 207  
*Kelly Easton, George Nemhauser and Michael Trick*  
1 Introduction 207  

2 Three Illustrative Examples 210  
2.1 The Generalized Assignment Problem 210  
2.2 The Traveling Tournament Problem 212  
2.3 The Social Golfers Problem 216  
2.4 Other Applications 218  

3 Implementation Issues 219  
3.1 Set Partitioning Versus Set Covering 219  
3.2 Initial Solution 220  
3.3 Column Management 221  
3.4 LP Relaxation 222  
3.5 Branching 223  
3.6 CP as a Subproblem Solver 224  
3.7 Column Generation Heuristics 224  
3.8 Combining Column and Row Generation 225  
3.9 Parallel Implementation Issues 226  

4 Future Directions for CP Based Branch-and-Price 226

References 229

8 Randomized Backtrack Search 233  
*Carla P. Gomes*  
1 Introduction 234
References

9

Local Search and Constraint Programming

LS and CP illustrated on a transportation Problem
Filippo Focacci, Francois Laburthe, Andrea Lodi

1 Introduction

2 A didactic transportation problem

3 A CP approach for dTP

3.1 A CP model for dTP

3.1.1 Basic model

3.2 Propagation

3.2.1 Disjunctive Relations

3.2.2 Linking trucks and bins

3.2.3 Propagating costs

3.3 A redundant routing model

3.3.1 Propagation

4 Constructive Algorithms

4.1 Insertion algorithms

4.2 Greedy insertion
<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3 Restricted Candidate Lists and GRASP</td>
</tr>
<tr>
<td>4.4 Discrepancy-based search</td>
</tr>
<tr>
<td>5 LS as Post-Optimization</td>
</tr>
<tr>
<td>5.1 LS + constraint checks</td>
</tr>
<tr>
<td>5.2 Constraint checks within the neighborhood iteration</td>
</tr>
<tr>
<td>5.3 Freezing Fragments</td>
</tr>
<tr>
<td>5.4 CP models for the Neighborhoods</td>
</tr>
<tr>
<td>6 Metaheuristics</td>
</tr>
<tr>
<td>6.1 Control strategies from metaheuristics</td>
</tr>
<tr>
<td>6.2 Managing several neighborhoods</td>
</tr>
<tr>
<td>7 LS during construction</td>
</tr>
<tr>
<td>7.1 Single Route Optimization</td>
</tr>
<tr>
<td>7.2 Ejection Chains</td>
</tr>
<tr>
<td>8 Conclusions</td>
</tr>
<tr>
<td>References</td>
</tr>
<tr>
<td>10 Open Perspectives</td>
</tr>
</tbody>
</table>

**Mark Wallace, Yves Caseau and Jean-Francois Puget**

1 Motivations, Challenges and Applications 331
1.1 Overview 331
1.2 Challenges 332
1.3 Focus on Hard to Solve Problems 333
1.3.1 Global Constraints 333
1.3.2 Scalability Limitations of Tree Search 334
1.3.3 Limitations of Global “Soft” Constraints 335
1.4 Problem Analysis vs. Resolution 335
1.4.1 Multi-criteria problems 335
1.4.2 Uncertain Problems 336
1.4.3 Probabilistic Data 337
1.5 Supporting the Problem Solving Process 337
1.6 Software Engineering Issues 339
1.6.1 Software Engineering in Support of Combinatorial Problem Solving 339
1.6.2 Combinatorial Problem Solving in Support of Software Engineering 340
2 Transforming Models to Algorithms 341
2.1 Conceptual and Design Models 341
2.2 Decompositions 342
2.3 Transformations 344
2.3.1 Separating Modeling for Performance Improvements 344
2.3.2 Motivations and Examples 345
2.3.3 Transforming Conceptual Models into Design Models 346
2.4 Search 347
2.4.1 Local and Constructive Search 347
2.4.2 Combining Different Search Methods 348
2.4.3 Concurrent Search 348
2.5 Inference 349
2.5.1 Global and Local Control of Inference 349
2.5.2 Controlling the Local Inference Procedure 350
2.5.3 Controlling Communication of Inferences 350
2.6 Symmetries 350

3 New Techniques 351
   3.1 Stochastic Optimisation 351
   3.2 Overconstrained problems and robustness 353
   3.3 User Support 354
   3.3.1 A Simple Environment for Solving LSICO Problems 354
   3.3.2 Automating Algorithm Development and Testing 356
   3.4 Packaging 357

4 New Application Areas 359
   4.1 Computer-Aided decision analysis based on simulation 359
   4.2 Cooperative Problem Solving 360
   4.3 Interleaved Planning and Execution 360

References 363

Index 367
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Interaction among constraints</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Cumulative constraint in discrete tomography</td>
<td>66</td>
</tr>
<tr>
<td>3.2</td>
<td>Two- and three-dimensional diffusion constraint in discrete tomography</td>
<td>66</td>
</tr>
<tr>
<td>3.3</td>
<td>Primitive constraints in CP</td>
<td>68</td>
</tr>
<tr>
<td>3.4</td>
<td>Primitive constraints in IP</td>
<td>69</td>
</tr>
<tr>
<td>3.5</td>
<td>Architecture for constraint-based solving</td>
<td>71</td>
</tr>
<tr>
<td>3.6</td>
<td>Inference in IP and CP</td>
<td>71</td>
</tr>
<tr>
<td>3.7</td>
<td>Supply Chain Example</td>
<td>78</td>
</tr>
<tr>
<td>4.1</td>
<td>An Assignment Timetable</td>
<td>92</td>
</tr>
<tr>
<td>4.2</td>
<td>An example of a global cardinality constraint</td>
<td>93</td>
</tr>
<tr>
<td>4.3</td>
<td>Rules of the ad-hoc filtering algorithm for the n-queens problem.</td>
<td>112</td>
</tr>
<tr>
<td>5.1</td>
<td>Linearization of all constraints in a single store</td>
<td>147</td>
</tr>
<tr>
<td>5.2</td>
<td>Optimization component in each constraint</td>
<td>148</td>
</tr>
<tr>
<td>5.3</td>
<td>Merging of two constraints</td>
<td>149</td>
</tr>
<tr>
<td>5.4</td>
<td>Global constraint architecture</td>
<td>157</td>
</tr>
<tr>
<td>5.5</td>
<td>Addition of cuts in optimization constraints</td>
<td>161</td>
</tr>
<tr>
<td>6.1</td>
<td>The input schedule and the PL cost function φ of an activity.</td>
<td>187</td>
</tr>
<tr>
<td>6.2</td>
<td>Results of running H(relax) and H(lambda) on the four test classes.</td>
<td>199</td>
</tr>
<tr>
<td>8.1</td>
<td>Two different executions of a backtrack search methods looking for an assignment to the variables A, B, C that satisfies the formula $(A \lor \neg B \lor \neg C) \land (B \lor \neg C) \land (A \lor C)$</td>
<td>238</td>
</tr>
<tr>
<td>8.2</td>
<td>Top panel: symmetric random walk (10,000 steps). Bottom panel: the heavy-tailed nature of a random walk contrasted with the exponential decay of the normal distribution.</td>
<td>243</td>
</tr>
</tbody>
</table>
8.3 Top panel: imbalanced tree model; $p$ is the probability that the branching heuristic will make a wrong decision, and $b$ is the branching factor (in this case, $b = 2$). This tree model has a finite mean and an infinite variance when $1/b^2 < p < 1/b$; both the mean and the variance are infinite when $p \geq 1/b$. Bottom panel: example distributions the imbalanced and bounded imbalanced models, contrasted with the balanced model. Parameters: $b = 2$, $n = 20$, $p = 0.5$ and $0.75$ ($n$ is the number of variables).

8.4 Quasigroup Completion Problem (QCP).

8.5 Erratic behavior of mean cost of completing a quasigroup (order 11, 30% pre-assignment) vs. stabilized behavior of mean for a standard distribution (gamma)

8.6 The phenomenon of long tails with randomized complete search procedures (QCP instance, order 11, 30% pre-assignment)

8.7 Left panel: log–log plot of heavy-tailed behavior (QCP instance, order 11, 30% pre-assignment; uncensored data). Right panel: no heavy-tailed behavior.

8.8 Log-log plot of heavy-tailed behavior.

8.9 Standardized normal, Cauchy, and Lévy densities

8.10 Comparison of the fatness of tails: normal (1,1), lognormal(1,1), exponential(1), and pareto(1,1).

8.11 Comparison of the tail fatness: normal (1,1), lognormal(1,1), exponential(1), and pareto(1,1).

8.12 Log-log plot of the complement to one of the cumulative distribution of the runtime for the round-robin sports scheduling (12 teams; CSP formulation); (a) full distribution; (b) tail ($X > 10,000$).

8.13 (a) Cost profiles for a logistics planning problems for depth-first and best-bound search strategies. (b) Heavy-tailed behavior of depth-first search.

8.14 Log-log plot of heavy-tailed behavior for SAT formulations of (a) logistics planning, and (b) register allocation

8.15 Log–log plot of the tail of the runtime distributions of an exact backtrack search coloring algorithm on three different graph topologies: random ($\log_2(p) = 0$); small-world ($\log_2(p) = -4$); and more structured ($\log_2(p) = -8$). $p$ is the probability of randomly rewiring an edge of a regular graph. Size of the regular graph: 100 nodes and degree 8 (Walsh, 1999).
List of Figures

8.16 Log–log plot of heavy-tailed behavior for the timetabling problem

8.17 Left panel: heavy tail for quasigroup completion with extensive propagation (CSP formulation, order 30, with 55% pre-assignment; uncensored data). Right panel: comparison of Satz to Relsat on logistics planning (SAT formulations).

8.18 Survival function \((1 - F(x))\) of the number of backtracks needed to solve different instances of model E with 20 variables and a domain size of 10. The parameter \(p\) captures the constrainedness of the instances; it is the probability of choosing one constraint (out of the total number of possible constraints). Two different regimes in the decay of the survival function can be identified: a heavy-tailed regime (curves with linear behavior) and a non-heavy-tailed regime.

8.19 Absence of heavy-tailed behavior for unsolvable QCP instances (CSP formulation)

8.20 Left panel: comparison of runtime profiles for proving optimality of misc07 from MIPLIB. Right panel: absence of heavy-tailed behavior in proving optimality of misc07

8.21 Restarts: (a) Tail (of \((1 - F(x))\)) as a function of the total number of backtracks for a QCP instance (CSP formulation), log–log scale; (b) same as (a), but for a logistics instance (SAT formulation). In (b) the left-most curve is for a cutoff value of 16, the middle curve is for a cutoff of 250, and the right-most curve is without restarts.

8.22 The effect of random restarts on solution cost for the logistics.d planning problem (SAT formulation)

8.23 Portfolio results for logistics planning: left panel, 2 processors; right panel, 20 processors

8.24 A range of portfolios for the MIP formulation of logistics planning (expected runtime).

8.25 Expected runtime (a) and expected total cost (b) of optimal portfolio for different numbers of processors.

9.1 Example of dTP with \(N = 10, M = 3, C = 10\).

9.2 Example of 3-opt move.

9.3 Example of node-transfer move.

9.4 Example of ejection chain.
List of Tables

3.1 Complexity of the polyomino reconstruction problem 63
3.2 Primitive and non-primitive constraints 68
3.3 Inference in IP and CP 70
6.1 Definition of the four classes of experiments. 198
8.1 Estimates of the index of stability ($\alpha$). $k$ is the sample size, and $u$ is the number of runs that failed to terminate before reaching the cutoff in the number of backtracks. The values within parentheses are the estimated asymptotic standard deviations. 252
8.2 Comparison of tail probabilities, $\Pr\{X > c\}$, for standard normal, Cauchy, and Lévy distributions 253
8.3 Estimates of the index of stability ($\alpha$). $k$ is the sample size. The values within parentheses are the estimated asymptotic standard deviations (SAT encoding). 261
8.4 Estimate of the index of stability ($\alpha$) for the school-timetabling problem. $k$ is the sample size. The value within parentheses is the estimated asymptotic standard deviation 263
8.5 Index of stability: impact of look-back strategies on heavy-tailed behavior (SAT encodings). 266
8.6 Solving (a) a 16-team round-robin scheduling problem (CSP formulation) and (b) the logistics.d instance (SAT formulation) for a range of cutoff values. 271
8.7 Restarts using a hybrid CSP/LP randomized backtrack search on a balanced QCP instance of order 50, at the phase transition. 272
8.8 Randomized rapid restarts (RRR) versus deterministic versions of backtrack search procedures (Satz solver used on SAT encodings; Ilog solver on CSP encodings). 275
During his invited talk at the first Workshop on Integration of Artificial Intelligence and Operations Research in Constraint Programming for Combinatorial Optimization, CP-AI-OR99, Jean-Francois Puget claimed that "Constraint Programming (CP) can be seen as Software Engineering applied to Operations Research (OR)". This statement is very strong, but indeed "almost" true. To my mind, CP is both something more and something less than this. It is something more since CP has its roots not only in Mathematical Programming, but also in Logic Programming and in Constraint Solving. From Logic Programming, CP inherits the declarative semantics, flexibility, the relational form which enables the definition of constraints without specifying input and output parameters, and the non-determinism which leads to the exploration of a search tree with a backtracking scheme. From Constraint Solving and Artificial Intelligence, CP has inherited powerful filtering algorithms, sophisticated search techniques and nogood recording. However, CP is something less than this: OR is so wide, and well studied since the fifties that so far CP could only cover a small, yet effective, portion of OR. The integration of areas of OR such as game theory, decision theory and stochastic programming into a CP framework has yet to be addressed.

However, experience with current implementations has already shown that CP is a powerful framework where Mathematical Programming concepts, and also local search techniques can be smoothly integrated and easily used.

The OR community has strengths both in breadth and in depth. The techniques of Integer and Linear programming and Local search are applicable to a huge swathe of (combinatorial) optimisation problems. Yet OR researchers have explored certain problem classes in great depth. These problems (the Travelling Salesman Problem, Set Partitioning and Covering, the Knapsack Problem, to name a few) are pure problems. No side constraints are considered. This simplification has an enormous advantage: pure problems are structured. The geometrical structure of the problem can help to reveal important problem properties. Geometric and algebraic structures enable practitioners to define ad hoc super-efficient (often polynomial) solving algorithms. Although pure problems rarely appear in real life applications, their investigation leads to the
development of algorithms that can be effectively exploited to solve variants of these problems. These algorithms can be considered as software components providing results exploited in a large variety of applications.

The exploitation of bounds, reduced costs, optimal solutions of subparts of the original problems, heuristic suggestions coming from relaxations or problem decompositions dramatically enhance CP solver performance. Linear programming solvers are nowadays integrated in all commercial CP solvers.

When it was first introduced, Constraint Logic Programming was a declarative, general framework for solving Constraint Satisfaction Problems. It was more efficient than Logic Programming, but for combinatorial optimization problems, it was not powerful enough to compete with OR methods.

Now, things have changed. CP is closer to software engineering applied to OR. Firstly global constraints have been introduced into CP solvers, embedding complex pruning techniques. They are a powerful, fundamental tool for modelling and solving constraint (optimization) problems. Secondly, the optimization side, that was naively treated at the beginning, is now becoming an important aspect of CP solvers exploiting OR techniques.

Many steps need to be taken. CP researchers need to study, learn and think to discover which is the most general way to incorporate OR components in CP solvers. It is a big challenge.

This book was conceived during the CP-AI-OR School on Optimization I organized in Le Croisic (France) in 2002. The School was a successful event with more than fifty participants, and the speakers covered almost all aspects of the integration in an exhaustive and stimulating way. After the school, I was solicited to collect papers concerning the talks into a book. Therefore, the book is mainly devoted to students, researchers and practitioners who are entering this evolving research field. The book covers a wide range of perspectives on the field. Beside chapters based on talks from the school, the book contains some additional papers covering aspects which were not treated during the school for reasons of time.

The book is organized as follows: Chapter 1 is devoted to an introduction that provides a high level overview of the main concepts of Constraint Programming (CP) and Integer Programming (IP) used in the book. Chapter 2 informally introduces integration methods describing and classifying the main works in the field. Chapter 3 presents a unifying framework that presents under a uniform perspective the main concepts of CP and IP, underlining similarities and differences and stating the basis for possible integrations. In Chapter 4 global constraints are described as a vehicle for integrating IP concepts in CP in a transparent way for the user. Chapter 5 presents various ways to integrate relaxations in Constraint Programming focussing on global constraints. Then, Chapter 6 describes hybrid solvers and Chapter 7 concerns Column Generation
and its integration in Constraint Programming. Chapter 8 concerns randomization and problem structure as a basis for understanding the intrinsic difficulty of the combinatorial problems. Many incomplete methods have been proposed mixing incomplete search (like local search and metaheuristics) in CP. Thus, Chapter 9 is devoted to a survey on the subject. Finally, the last chapter is devoted to open perspectives and future directions.

The authors are eminent and well known researchers and have significantly contributed to the field. They come from Universities, Research Centers and industries.

Finally, I would like to thank all the people who have helped me in the realization of this book. Andrea Lodi and Jean-Francois Puget suggested me to write this book. Krzysztof Apt spared his precious time to help me at the beginning of the process with invaluable advice. Mark Wallace who kindly proofread some parts of this book. John Hooker introduced me to Kluwer and pointed me out the CS/OR series. Gary Folven provided me with constant assistance. François Laburthe and Narendra Jussien helped me in the organization of the School of Optimization in Le Croisic.

Warm thanks go to the authors of the papers presented here for their work and their constant commitment in respecting the numerous requests I have asked of them, among which were requests to review other chapters in the book. Besides the authors, I would like to thank external reviewers: Torsten Fahle, Marco Gavanelli, Willem Jan van Hoeve, Ulrich Junker, Meinolf Sellman and Louis Martin Rousseau. I would also like to thank my colleagues Luca Benini and Andrea Roli for their help in the editing of several parts of the manuscript.

MICHELA MILANO
Contributing Authors

Farid Ajili
IC-Parc, Imperial College London
London SW7 2AZ, United Kingdom.
f.ajili@imperial.ac.uk

Alexander Bockmayr
Université Henri Poincaré, LORIA, B.P. 239
54506 Vandœuvre-lès-Nancy, France
bockmayr@loria.fr

Yves Caseau
BOUYGUES TELECOM
20, quai du Point du Jour
92640 Boulogne Billancourt, France
yves@caseau.com

Emilie Danna
ILOG and LIA,
Université d’Avignon,
CNRS-FRE 2487
edanna@ilog.fr

Kelly King Easton
Department of Industrial and Manufacturing Engineering
Kansas State University
237 Durland Hall - Manhattan, KS 66506
keaston@isye.gatech.edu

Filippo Focacci
ILOG S.A.