Preface

Physics is like sex: sure, it may give some practical results, but that's not why we do it.
— Richard P. Feynman

Judge a man by his questions rather than his answers.
— Voltaire

Scientists investigate that which already is.
Engineers create that which has never been.
— Albert Einstein

This book is an introduction to constraint-based local search, a novel way to solve combinatorial optimization problems by local search.

Combinatorial Optimization

Combinatorial optimization problems are ubiquitous in our society. From the transportation industry to supply-chain management, from manufacturing to the pharmaceutical industry, from resource allocation to sports scheduling, they affect almost every aspect of our daily lives. Over the last decades, optimization technology has progressed rapidly and has found its way into novel application areas, helping society to address critical situations and industries to become more reactive and cost-effective. You can be sure that as you are reading this sentence someone, someplace, is solving an optimization problem for some practical purpose.

In general, optimization problems are extremely challenging computationally. They cannot be solved exactly in polynomial time and no single approach is likely to be effective on all problems, or even on all instances of a single problem. Solving optimization problems remains a very experimental endeavor: what will or will not work in practice is hard to predict. Some problems are better solved using mathematical programming, some are more amenable to solutions by constraint programming, while local search is more effective on others.

In addition to their inherent computational complexity, algorithms for solving optimization problems are often large and intricate: they are tedious to design, implement, and maintain. This is ironic, of course, because many optimization problems can be specified concisely and declaratively. The considerable distance between the specification and the final program indicates that programming languages are seriously lacking in expressiveness and abstractions in this application area. As a consequence, it is not surprising that modeling and programming languages have been designed to narrow this conceptual gap.

Optimization Tools

Languages for expressing combinatorial optimization problems have historically focused on integer programming. Typically, integer programs are solved using branch and bound (or branch and cut), a systematic search procedure using linear relaxations to prune suboptimal nodes. Starting from matrix generators, this line of research led to the development of elegant modeling languages (for instance, [11, 32]), where collections of (typically) linear constraints are expressed using traditional algebraic and set notations. These constraints, and the associated objective function, are then sent to integer programming solvers.

The last two decades have witnessed the emergence of *constraint programming* as a fundamental methodology for solving a variety of combinatorial applications, and of *constraint programming languages* to express them (for instance, [53, 120, 121]). The focus of constraint programming is on reducing the search space by pruning values that cannot appear in any feasible or optimal solution. Because of the distinct nature of this paradigm, it is not surprising that constraint programming languages have made many innovations in the modeling of combinatorial optimization problems. Typically, constraint programming languages feature rich languages for expressing and combining constraints and for specifying search procedures at a high level of abstraction. These languages support the fundamental concept of combinatorial constraints (for instance, [9, 120, 97]), which capture combinatorial substructures arising in many applications. Moreover, there have been several attempts to unify constraint programming and mathematical modeling languages (for instance, [33, 121, 122]).

Local Search

Local search approaches the solving of combinatorial optimization problems from a very different angle from the systematic tree search of constraint and integer programming. From a theoretical standpoint, local search algorithms sacrifice quality guarantees for performance. They may fail to find optimal, or even high-quality, solutions. However, on many problems, they isolate optimal or near-optimal solutions within very reasonable time constraints. From a computational standpoint, local search explores a graph, moving from solutions to neighboring solutions in the hope of improving the value of the objective function. Defining this neighborhood graph and exploring it effectively are two of the main issues faced by local search algorithms.

Local search is particularly appropriate for large-scale problems involving thousands of decision variables and for online optimization problems in which (good) solutions must be found within strict time constraints. Local search is also the technique of choice in many optimization applications, where systematic search techniques are less effective. For instance, at the time of writing, the best approach to the traveling tournament problem, an abstraction of major-league baseball scheduling, is a neighborhood-search method that significantly outperforms constraint and mathematical pro-

gramming. The same is true of many other important problems, such as vehicle routing, frequency allocation, and many resource-allocation and scheduling problems.

Constraint-Based Local Search

Surprisingly, the modeling and programming language communities have largely ignored local search. Part of the reason is that local search algorithms are often presented in terms of low-level (implementation) concepts. Scientific papers rarely specify local search algorithms using constraints and objective functions. Rather they talk about neighborhoods, local moves, and metaheuristics either informally or using low-level implementation details. Such papers seem to offer little opportunity for reuse, separation of concerns, and modularity. Moreover, the wealth of research on metaheuristics further exacerbates the perception that local search is a collection of heterogeneous techniques lacking a unifying theme.

But local search was too important a paradigm for combinatorial optimization to be ignored indefinitely. The 1990s witnessed significant progress in solving satisfiability and integer programming problems by local search (for instance, [105, 107, 135]). LOCALIZER [70, 72], the first modeling language for local search, introduced the use of invariants to specify incremental algorithms declaratively. By the beginning of the 21st century, combinatorial constraints were also recognized as beneficial in local search (for instance, [16, 36, 73, 84]).

Constraint-based local search, the idea of using constraints to describe and control local search, was slowly emerging. The COMET project was initiated in 2001 to explore the ramifications of constraint-based local search and how it could be supported in high-level programming languages. The project led to a novel constraint-based architecture for local search [74], new modeling and control abstractions [124, 125, 128], and the programming language COMET.

Comet

COMET is an object-oriented language with a number of innovative modeling and control abstractions for local search.

From a language standpoint, COMET supports both modeling and search abstractions in the spirit of constraint programming. The modeling abstractions feature invariants and a rich constraint language, that includes numerical, logical, and combinatorial constraints as well as constraint combinators. The search abstractions include randomized selectors, events, checkpoints, neighbors, and novel control structures to implement nondeterminism. Many of these control abstractions rely on first-class closures and continuations as a unifying implementation technology. COMET is also an open language: programmers can add their own constraints and objectives, as well as their own control abstractions.

From a computational standpoint, COMET is particularly innovative. Constraints and objective functions are differentiable objects that maintain properties that are then used to direct the graph exploration. Moreover, *differentiable objects* can be queried to determine the impact of local moves on their properties. In particular, constraints are differentiable objects maintaining their violations, while objectives maintain their evaluation. These constraints and objectives can then be naturally composed in combinators to build more complex modeling objects. Typically, differentiable objects the encapsulate efficient incremental algorithms that are so fundamental in obtaining high performance in most local search algorithms.

Benefits

The constraint-based architecture of COMET brings numerous benefits in solving combinatorial optimization problems by local search. Most of these benefits are expressed in the formula

$$Local\ Search = Model + Search$$

stating that local search algorithms can be specified in terms of modeling and search components. The modeling component, which is purely declarative, expresses the combinatorial structure of the application in terms of constraints and objective functions. The search component exploits the structure expressed in the model to guide the neighborhood exploration toward high-quality solutions.

A Rich Modeling Language for Local Search Perhaps the most significant benefit of constraint-based local search is its rich modeling language. As mentioned earlier, local search algorithms are too often described in terms of operational concepts. The constraint-based architecture of COMET allows the same applications to be described declaratively in terms of constraints and objectives that specify properties of the solutions. This model captures the combinatorial structure of the application and is largely technology-independent. COMET models are frequently similar to constraint-programming models for the same applications. As a result, constraint-based local search moves the development closer to modeling, decreasing the distance between the application and the computer program.

Moreover, models in COMET are built compositionally, by combining constraints and objects through a wide variety of combinators. Constraints and objectives can be combined through arithmetic, logical, and cardinality operators. In practice, compositionality is fundamental to expressing the complex idiosyncratic constraints that typically arise in industrial applications, accommodating the changes often requested by customers over the course of a project, and capturing the combinatorial structure of the application.

A Rich Search Language for Local Search Equally important is the rich search language of COMET, which abstracts many of the tedious and error-prone aspects of local search algorithms.

COMET supports abstractions such as solutions and selectors to ease the formulation of heuristics, events to glue together different aspects of the search procedure, the concept of neighbor to address the temporal disconnection between move specifications and executions in complex neighborhoods, and nondeterministic abstractions that are becoming increasingly important with the progress in hybridizations of local and systematic search. The control abstractions of COMET promote modularity and reuse. In particular, they make it possible to separate, to a large extent, heuristics from metaheuristics, neighborhood definitions from their uses, and search heuristics from search strategies.

Separation of Model and Search The formula clearly highlights another fundamental benefit of the architecture: the clean separation between model and search. Although the model and the search often interact in sophisticated ways, they are physically separated in the COMET code. Moreover, the model and the search procedure can evolve independently: the search procedure can be replaced without affecting the model and the model can be enhanced without upgrading the search procedure.

The separation of model and search introduces another desirable benefit: generic search procedures that can be reused in many contexts. Indeed, since models and search procedures interact only through limited channels, the search algorithms are often independent of the specifics of the models.

Extensibility and Flexibility Another interesting property of COMET is the extensibility of its architecture, both for modeling and search.

At the model level, new constraints and objectives can be defined in the language itself and combined exactly as other differentiable objects. The implementation of these differentiable objects is greatly simplified by the availability of invariants that provide an intermediate layer between the language and differentiable objects. Invariants, or one-way constraints, maintain (possibly complex) algebraic, set, and graph expressions incrementally under changes in the decision variablesm, and often simplify the implementation of differentiable objects significantly.

At the search level, the availability of closures and continuations as the enabling technology new control structures possible that simplify the specification of complex search algorithms. Similarly, the nondeterministic instructions are parameterized by search controllers, making it easy to specify new search strategies and new implementation schemes.

It is important to emphasize an important property of constraint-based local search: constraints (or objectives) are independent of each other and interact only through incremental variables. The resulting flexibility greatly simplifies the definition of new constraints and objectives since the differentiable objects can be implemented in isolation, and makes it easy to add constraints in a model without affecting the rest of the model and the search.

Efficiency COMET's rich modeling language and concept of differentiable objects often make it comparable in efficiency to low-level implementations, especially for complex applications and for problems with idiosyncratic constraints. Part of the reason is the fact that the modeling abstractions and combinatorial objects encapsulate sophisticated incremental algorithms that exploit their underlying structure.

New Perspectives Finally, it is important to mention the long-term implications of constraint-based local search. Its formula

$$Local\ \ Search = Model + Search$$

directly parallels the constraint programming expression

$$Constraint\ \ Programming = Constraints + Search$$

essentially indicating that local search and constraint programming solutions can now be expressed at a similar level of abstraction. Moreover, the local search and constraint programming models are often similar in nature, and integer programs can also be derived systematically from them. As a result, it is natural to envision modeling systems that integrate these orthogonal technologies in a unique platform in which experimenting with all of them, as well as their hybridizations, would be easy. Although considerable research is required to lay the foundations of such modeling systems, constraint-based local search highlights the importance of high-level models that convey the structure of applications.

Contents

This book is an introduction to constraint-based local search and its implementation in COMET. Part I gives an overview of local search, including neighborhoods, heuristics, and metaheuristics. It is not intended to be comprehensive, since there exist excellent textbooks covering local search, as well as portions of each of these topics. Rather, this part aims at presenting, in a unifying way, the main concepts used throughout the book. Part II presents constraint-based local search, its architecture, and its modeling and search components. Part III describes how constraint-based local search is supported in COMET and cover invariants, differentiable objects, control, and first-order control. Part IV describes a variety of applications to illustrate constraint-based local and COMET. The clustering of applications by metaheuristics provides a clear focus for each chapter. Part V presents scheduling applications which are particularly challenging. Indeed, they often feature implicit neighborhoods and require additional modeling support to convey their combinatorial structure naturally. This part gives the necessary background on scheduling, describes the novel vertical abstractions, and considers various applications in job-shop and cumulative scheduling.

This book illustrates constraint-based local search on a number of satisfiability problems, in which the goal is to find a feasible solution, not an optimal one. There are many such applications in practice, and constraint-based local search is often a natural vehicle to model and solve them. They illustrate an additional strength of constraint-based local search: its ability to cope with both satisfiability and optimization problems in a uniform fashion.

Disclaimers

This book lies at the intersection of several fields, it uses an actual programming language as a proof of concept, and it discusses a variety of applications. It is thus important to provide the proper context to understand the terminology, to interpret the experimental results, and to evaluate the models.

Terminology Combinatorial optimization bridges computer science, industrial engineering, and operations research: this makes the area stimulating but complicates the exchange of ideas because of differences in vocabulary. This book adopts the following conventions for the main underlying concepts. A *solution* is an assignment of values to the decision variables. A *feasible solution* is a solution satisfying all constraints. An *optimal solution* is a feasible solution optimizing the value of the objective solution.

Efficiency The book contains experimental results for a variety of applications. These results aim merely at suggesting the practicability of constraint-based local search. They are certainly not indicative of the best performance that can be achieved in these applications, either by constraint-based local search and by low-level implementations of the same algorithms. In addition, the experimental results are not always scientifically rigorous: they may be limited to a small number of instances on some problems, and in this case they are provided only to give some indication of the algorithm behavior and potential. In general, the references give more information about performance.

COMET and constraint-based local search in general introduce several levels of indirection compared to low-level implementations. For some (pure) applications in which the basic operations are extremely simple, the induced (constant) overhead may be significant if very high performance is desired. Typically, however, for complex applications involving heterogeneous and idiosyncratic constraints and complex neighborhoods, COMET's overhead is small. It may become negligible or may be completely offset by the reduction in development time or ease in experimentation. These are precisely COMET's target applications: ones that require considerable development time and experimentation, which is greatly simplified by constraint-based local search.

It is also important to mention that COMET does not yet include all the traditional low-level optimizations found in high-performance compilers, although its implementation is based on a just-in-

time (JIT) compiler that generates machine code. There is thus considerable room for improvement and one may expect future implementations of constraint-based local search to be more efficient and to reduce the gap from low-level languages.

Finally, it is worth emphasizing that constraint-based local search is independent of its implementation technology. It can be supported in any programming language or implemented as an object-oriented library. This is especially true of the modeling aspects of constraint-based local search: the invariants and the differentiable objects. The control abstractions are more challenging to support elegantly in modern languages, which often lack the low-level implementation technology: closures and continuations.

Models and Algorithms The models and algorithms presented in this book are merely illustrative and are not necessarily the best possible algorithms for some of the applications. In general, they are included because they illustrate a new feature of constraint-based local search or because they show how to implement certain classes of local search algorithms in COMET. They should certainly not be seen as promoting one approach over another. Indeed, this would contradict one of the main motivations behind this book: to show that many of these approaches have orthogonal strengths and that it is difficult to predict in advance what will or will not work on a particular instance of a specific problem.

Notes and Further Reading The *Notes and Further Reading* in most chapters direct readers to references where additional information may be found. They are not intended to be exhaustive. Instead, they simply provide the starting links for search engines and long random walks in interesting neighborhoods.

Syntax and Semantics COMET is an ongoing project: the language and its implementation are likely to evolve over the coming years. In particular, its syntax and semantics may change, so that some programs presented here may not run on subsequent versions or may give different results. More information about the language, its implementation, its distribution, and example programs is available at `<www.comet-online.org>`.

Acknowledgments

The research underlying this book can be traced back to early 1994 when Paris Kanellakis asked the authors "whether constraint programming languages could accommodate local search." It took us more than 10 years to come up with a long and late answer to this far-reaching question. This book gives us a wonderful opportunity to acknowledge Paris's scientific vision and engaging personality.

Of course, this research did not take place in a vacuum. It builds on the work of many colleagues who are, we hope, properly acknowledged in the text. In particular, many COMET applications have

been directly inspired by local search algorithms proposed by other researchers. We also would like to thank the undergraduate and graduate students at Brown University and the University of Connecticut who contributed to this project, including Aris Anagnostopoulos, Ionut Aron, Russell Bent, Ivan Dotu, Daniel Fontaine, Greg Harm, Liyuan Liu, Keith Schmidt, and Yannis Vergados. Each of them improved our understanding of this material in more ways than they can imagine.

We would not even have thought of starting this project without knowing that Cambridge, MA, has a Legal Seafood restaurant where Bob Prior patiently listens to passionate scientists, rewrites their ideas on place mats, and makes them believe that they have, or sometimes don't have, a story to tell. May be one day, Bob will go back to all these place mats, write a book about all the strange characters he meets, and explain his secret recipes.

We were also fortunate to have amazing editors for the manuscript: Katrina Avery, Deborah Cantor-Adams, and Elisabeth Nelson helped to improve the manuscript in many ways, even correcting some of the formulas and programs and suggesting better quotes.

Last but not least, there are two wonderful women and four equally wonderful children who are eternally grateful to Bob for ensuring that we can all spend Christmas together, playing football or knights and princesses, singing, or simply talking around Belgian food and French wine in a computer-free environment.

Pascal Van Hentenryck

Laurent Michel