Constraint Programming Letters 1 (2007) 21-32 Submitted 5/2007; Published 11/2007

Good and Bad Futures for Constraint Programming (and Operations Research)

John N. Hooker JOHN@HOOKER.TEPPER.CMU.EDU
Tepper School of Business
Carnegie Mellon University
Pittsburgh, USA

Editor: Pascal Van Hentenryck

Abstract

Two futures are sketched for constraint programming and operations research. In one, they continue their present emphasis on computational methods. In the other, they are empirical sciences dedicated to prescriptive modeling of human activities, with computation playing an ancillary role. The second future is defended as one in which the two fields, which are at root one field, maintain their vitality and make a more effective contribution to solving the problems of an increasingly complex world.

1. Introduction

The ideal future for constraint programming (CP) is to pursue what I will argue is its core purpose: prescriptive modeling. A less desirable future is for it to continue its current focus on computational methods.

CP has an interest in modeling that dates back to its origins in logic programming. Yet most of the serious academic work in the field deals with solution methods. Operations research (OR) has followed a similar pattern. Neither field views modeling as its core scientific mission. I wish to make a case for precisely the opposite. Modeling is what a scientific discipline called CP or OR should be all about.

This claim may seem implausible because no real science has grown up around the modeling task. Algorithms can be analyzed with powerful mathematical tools, but modeling is more an art than a science. Yet this comparison confuses the creation of a model with its analysis. No science has grown up around the task of writing an algorithm, either. It is no less an art than writing an model. Formulating an algorithm or a model is an irreducibly creative act. Science enters the picture when one analyzes the algorithm or model, as when a computer scientist analyzes the complexity of an algorithm.

A scientific discipline can therefore focus on either algorithms or models. Computer science is concerned with algorithms, while most of the sciences (e.g., physics, biology, cosmology) create models of some phenomenon. The essence of CP and OR, I will argue, is to create and analyze a particular type of model: a prescriptive model of human activity.

2. Taking the Modeling Task Seriously

We know what a scientific model looks like. In physics, for example, it is thermodynamics, statistical mechanics, or the big bang theory. It is a succinct description of some natural phenomenon, usually written in a formal language, from which one can rigorously deduce empirically testable consequences.

Where can we find models of this kind in CP or OR? I grant that many of our models are not like this, because we have forgotten what models are supposed to do. We equate "model" with "problem formulation," when it is actually much more than this. A model should be explanatory as well as predictive. Its structure should elucidate the structure of the phenomenon it describes.

The situation is particularly bad in OR. A model in this field is too often a long list of undifferentiated inequality constraints, perhaps thousands of them. The model is certainly predictive, because one can rigorously deduce consequences (solutions) that can be tested against real experience. But it is anything but explanatory.

CP is in somewhat better shape due to its custom of using global constraints. A global constraint represents an island of structure in the problem. If a problem can be formulated in terms of a few global constraints, then the structure of the model mirrors the structure of the problem to some extent and helps to clarify our understanding of it.

Even in OR, however, one can find instances of true scientific modeling. Network flow models provide a textbook case (Bazaraa et al., 2004). A network flow model fits a wide range of problems and makes them easier to think about. It elucidates the problem rather than obscuring it behind a long list of inequalities. Efficient algorithms allow one to deduce feasible and optimal solutions from the model. One can implement these solutions to learn whether they deliver the results predicted by the model.

An even clearer example is provided by critical path models for project scheduling, which have been used worldwide for decades. They are based on the very simple idea that jobs on the longest path in a precedence network are critical to finishing the project on time (Kelley, 1961). These models not only allow us to compute schedules, but the idea of a critical path helps us to understand why projects get behind and what to do about it.

Consider edge-finding rules for scheduling, which have key applications in both OR and CP (Carlier, 1982; Carlier and Pinson, 1989). A single-machine scheduling problem with time windows can be viewed as a natural phenomenon that we would like to understand. When we learn how to deduce precedence relations between the jobs (edge finding), we learn something about the structure of the problem. The edge-finding rules explain why certain schedules are possible are others are not, much as statistical mechanics explains why a hot gas expands.

Similarly, the cumulative scheduling algorithms developed in the CP community have explanatory value. These algorithms reason about the "energy" required by a job in order to derive edge-finding, extended edge-finding, not-first/not-last, and similar rules for reducing variable domains (Baptiste et al., 2001). They not only compute feasible schedules but tell us that energy-based reasoning is key to understanding cumulative scheduling. To my knowledge, this kind of reasoning has not become part of the everyday discourse of schedulers in the way that critical path reasoning has, but it could and perhaps should,

because it could give them an intellectual framework for a deeper understanding of their task.

The central point here is that physics cannot explain scheduling in the way that CP or OR explains it. It uses the wrong level of analysis. It can tell us why the machines work, but it cannot tell us why some schedules are possible and others or not. Computer science cannot explain this, either. It can tell us about the complexity of edge-finding algorithms, but this is a different matter. CP and OR alone offer the right kind of theory—the right level of analysis—to elucidate scheduling as such. This is why they should take the modeling task seriously.

There is no theory to guide CP or OR modeling, but this is true of any field. Formulating a general theory of relativity is a supremely creative act. There is no theory of modeling because theories *are* models. Science enters with the analysis and verification of a model, and it is no different with CP or OR models. They should be treated no less as scientific models than Ludwig Boltzmann treated statistical mechanics.

3. The Subject Matter of CP Models

It is not enough to say that CP creates and analyzes scientific models. To characterize the field, I must say something about the subject matter of its models. If chemistry studies molecules and biology studies living things, what does CP study?

I think it studies the combinatorial structure of human activities. The activities include scheduling, production planning, logistics, product design, and countless others. CP is not the only field that studies combinatorial structure. For example, combinatorics studies combinatorial structure. The difference is that combinatorics does not focus on human activities. It thinks at a greater level of abstraction, which gives it more generality but fewer results. It can help analyze a CP problem, much as the theory of differential equations can help analyze heat transfer. But an understanding of heat transfer requires more than the theory of differential equations, just as machine scheduling requires more than classical combinatorics.

The word "combinatorial" may be a little too narrow, and it is certainly too narrow for OR. For lack of a better term, one might more generally say that CP and OR study the mathematical structure of human activities. But this is not to say that CP or OR is mathematics, applied mathematics, or anything of the sort. CP and OR are not mathematics because mathematics, like its subfield combinatorics, operates on a level that is too general to understand the structure of human activities. CP and OR are not applied mathematics, either, because they require more than applying mathematical theory and techniques to human activities. CP and OR require theory formation of their own: they study the structure of scheduling, planning, logistics, and product design, and not just directed graphs, polyhedra, and other mathematical objects.

I may step on some toes at this point, since many people in CP and OR regard themselves as mathematicians, particularly mathematical programmers. But this is quite consistent with what I am saying. CP and OR people may very well discover that existing mathematical tools require further development to meet the needs of their field, just as a physicist may discover the same for quantum mechanics. To the extent that they develop the tools

themselves, they are mathematicians. But this does not mean that CP and OR are branches of mathematics.

The case of mathematical programming is instructive. The term derives from "linear programming," which is George Dantzig's name for the formulation of linear optimization models for logistics problems (Dantzig (1991) tells us that Robert Dorfman later suggested the term "mathematical programming" to cover both linear and nonlinear models). The key point is that the formulation, analysis, and solution of linear programming models for logistics is OR, not mathematics, because it helps us to understand the structure of logistics problems. It is true that the analysis and solution required further development of such existing mathematical ideas as linear algebra, the Farkas Lemma, and so forth. To the extent that this development created a theory of linear optimization in the abstract, it can be regarded as a contribution to mathematics as well as to OR. In other words, mathematical programming is in fact a branch of mathematics, and OR people regularly contribute to it.

Yet Dantzig's signal contribution was not just a mathematical result, but the whole package: the discovery that a wide class of human activities (much wider than his logistics problems, it turns out) have a linear structure that can be analyzed in an interesting and useful way. The explanatory value of this discovery is remarkable. It explains why many activities tend to operate a zero level in good operational plans (because many variables are nonbasic at an optimal vertex solution). It explains exactly why the best plan is in fact the best (because the nonbasic activities have nonnegative reduced costs). It explains how changes in resource availability affect the cost of the operation (it depends on the value of dual variables, which become shadow prices). It even gives us insight into two-person noncooperative games (since the two players are modeled by the linear problem and its dual). This is much more than the mathematics of linear programming, which explains how polyhedra relate to linear functionals. It is OR at its best.

Computer science has a similar relation to CP and OR. Some people in these fields regard themselves as computer scientists, and they in fact contribute to the discipline. But computer science is fundamentally about understanding the nature of computation rather than understanding the structure of human activities. The insights and techniques developed in this process contribute to CP/OR, but they do not comprise CP/OR.

4. The Role of Computation

If modeling rather than computation is to lie at the core of CP and OR, how do I explain our preoccupation with computational issues? Computation must surely have some legitimate role in these fields.

Computation has much the same function in CP and OR as in physics and chemistry. It is the way we deduce the consequences of a model. Solving Schrödinger's equation is a daunting task even for the simplest atoms, but a crucial one if we are to know what sort of atomic structure is predicted by quantum theory. Computation is equally indispensable

to CP and OR modeling. Linear programming, for example, owes its success partly to the fact that electronic computers became available shortly after the theory was developed.[1]

While indispensable, computation should nonetheless be servant rather than master. Just as relativity physics does not exist to develop Riemannian geometry or quantum mechanics to develop group theory, CP does not exist to develop domain filtering methods. Rather, filtering methods exist to help us understand the consequences of CP models.

Certainly the analysis of model structure is intellectually related to the computational problem, since this analysis gives us clues as to how to write algorithms to exploit the structure. We are likely to address the computational problem when we develop and analyze models. Conversely, efforts to write fast algorithms sensitize us to problem structure.

CP is aware of this interaction, but primarily in only one direction. It analyzes global constraints to exploit structure for computational purposes, and the better models come only as a byproduct. The tail wags the dog. Computational methodology is important, but it should be viewed as ancillary to the field rather than its intellectual core.

It may, of course, be possible to develop a mathematical subfield that is to CP what mathematical programming is to OR. It would study a mathematical object, perhaps a constraint satisfaction problem. It would develop solution methods and have its own literature. I am not sure whether there is material here to justify a branch of mathematics. A mathematical field, like any other, should be defined by a desire to understand a phenomenon, whether it be numbers or polytopes or higher dimensional space, and not by a collection of techniques. In any case, it would be a mistake to identify such a subfield with CP. It would be like identifying OR with mathematical programming. CP is vitally concerned with understanding human activities and helping people involved in them to make decisions, and none of this would be directly addressed by a purely mathematical field.

5. An Empirical Science?

I am suggesting that CP and OR are empirical sciences, no less than physics or chemistry. One can implement the solution of a model and check whether the outcome is as predicted by the model. If the prediction is incorrect, the model is false and must be rejected. A scheduling model, for example, is incorrect if no one can implement a schedule that the model says is feasible.

This idea may seem wrong-headed because we never speak of a model as being "false." If a feasible solution never occurs in practice, we simply applied the wrong model. The model itself may be perfectly all right for another problem. It therefore makes no sense to say the model is false, as physicists said of Newton's Laws when some observations disconfirmed them.

Yet the comparison with Newton's Laws is misleading. They are intended as a model, or rather part of a model, for the behavior of matter and energy throughout the universe. This is why they must be rejected if they are wrong even once. A CP model is more like a model of an ecosystem. A model of the Indian Ocean, for example, might track convection currents, fish migration, nutrient balance, and so forth. If it fails to predict a major decline

1. The first commercial linear programming model was written by Alan Manne and William Cooper in the basement of the building where I work, in part because it housed one of the world's few computers at the time.

in fish stocks, then it must be revised or rejected. The model is in fact false, since one judges a model by what it is supposed to model. If we fail to speak of CP models as true or false, then perhaps this is because we, again, have forgotten what models are supposed to do.

My claim is that a CP model, like a physics or chemistry model, is intellectually interesting to the extent that it describes and explains what happens out there in the world. Its evaluation is therefore incomplete until it is confirmed empirically. There are, however, various ways to do this. One way is to check whether the models "assumptions" hold, since they in some sense represent the model's empirical content. I write the word in quotes because we should not actually *assume* anything. The "assumptions" are assertions about what is really happening and should be verified as such.

Suppose for example that we formulate the classical single-machine nonpreemptive scheduling model with time windows. The model carries the claim that it explains and predicts the behavior of certain real scheduling activities. In particular, it predicts that certain schedules are possible. To test this prediction, it may not be necessary to go into a factory and try to implement these schedules to see if they are really possible. If we can verify that the model's "assumptions" hold, then we know that these schedules are possible. Some of these assumptions may be easy to check. For example, we can check that only one job is underway at any given tine, or that a job finishes without interruption once it has started.

Other assumptions, however, are not so easy to verify, because they require proving a negative. We must do more than verify that every constraint is valid; we must verify that *no other constraints* are valid. So while we may be able to perform some degree of empirical validation from an armchair, full testing requires greater familiarity with the phenomenon. It is not so much a matter of taking measurements, as one would do in physics, but a matter of truly accounting for the combinatorial features of the problem. If the model makes incorrect predictions, then perhaps the real situation has some structural elements that the model overlooks. Perhaps the setup times are sequence-dependent, for example, a fact that calls for a substantially different model. Empirical verification is a matter of making sure that the model adequately reflects the structural features of the real-world problem.

6. Prescriptive Modeling

There is a major difference between CP/OR and the other empirical sciences. CP and OR aim not only to explain phenomena but to guide decision making. Some of the variables in its models represent human choices rather than the state of nature. The models do not predict human choices but predict the consequences of those choices. This makes the models *prescriptive* as well as descriptive.

Perhaps it is the prescriptive nature of CP/OR models that, above all else, has led these fields away from explanatory modeling. The overriding objective has been to give advice to the client. For this purpose it may seem to matter little whether the model looks neat, so along as it yields some useful numbers. This attitude is understandable, but it carries a high cost.

The most immediate cost is that we have sidelined the development of what OR people call postoptimality analysis. It has a modest following in OR, and there is a related if small literature on explanation in CP. Yet neither has ever been a mainstream research activity, and little of what we know has been implemented in the major commercial solvers. As a result we spend substantial effort developing models but extract relatively little information from them, perhaps only a single solution.

More generally, our models give us numbers but limited insight. It is not as though no one is aware of this. Thirty years ago, Art Geoffrion, one of the seminal figures of OR, wrote a famous article entitled, "The purpose of mathematical programming is insight, not numbers" (Goeffrion, 1976). Nonetheless our models continue to be regarded primarily as black boxes that spit out numbers. There are countless anecdotes about clients who do not trust the recommendations of a number-crunching consultant because they do not know what is in the black box.

Explanation and elucidation are every bit as important for prescriptive as for descriptive modeling, perhaps more so, because models recommend actions that may incur substantial cost and risk. It is unreasonable to ask decision makers to trust the computer. They must understand, in their own minds, why a certain course of action is better. Explanatory models provide insights that can be conveyed to decision makers.

7. A Bad Future

I have proposed a good future for CP and OR. A bad future is one in which they are defined by their computational techniques rather than the phenomena they study. This kind of future is bad for both practical and theoretical reasons.

On the practical side, a discipline defined by its techniques tends to become resistant to new techniques, and in the process loses vitality and relevance. I have been told that something like this happened to control theory. It identified itself with certain classical methods and resisted new techniques developed for robotics. As a result the robotics community went its own way and took with it an exciting set of problems. A similarly negative reaction to fuzzy control sent that community in a different direction. Control theory saw new techniques as threatening, because any research activity that introduced new techniques was, by definition, not control theory. It should have identified itself with the problem of control and enthusiastically embraced any new methods that addressed the problem from a different angle.

One may argue that OR has remained strong and relevant for fifty years, despite the fact that it is, to a large degree, a bundle of techniques. Yet the field is not purely technique. There is a core of explanatory modeling, such as the linear programming model just described, and this may account for much of the field's longevity. At the same time there is periodic hand-wringing in OR about its future, the declining membership of professional societies, the near-disappearance of OR from MBA programs, and the uncertain place of OR in the university (there are very few "Departments of OR") (Churchman, 1970; Gass, 1987; Horner, 2003; Samuelson, 1996; Trick, 2002). True, the field repeatedly outlives pronouncements that "OR is dead" (Ackoff, 1979; Corbett and van Wassenhove, 1993) and it continues to score many successes in both theory and practice. Yet perennial concerns

about the status of the field do not come from nothing. I never hear anyone agonize about the future of computer science, for example, or its legitimacy in the academy.

This brings to me a second practical problem. Since OR has not clearly identified a phenomenon it wants to study, nobody really knows what OR is. There is no consensus on how exactly OR differs from management science.[2] Our academic colleagues do not know where to put us, because they honestly do not understand what we do. OR is wedged into engineering departments, business schools, mathematical sciences departments, and even statistics departments. CP has a similar problem. It is typically a corner of the computer science department, even though, if I am right, it is a very different field than computer science and should have its own legitimacy.

The problem extends beyond the academy. I learned long ago never to tell people that I do operations research, because they conclude that I am a surgeon. Few people have ever heard of OR, even though it has existed for over fifty years, and we encounter OR frequently in everyday life—when we make an airline reservation (yield management), buy consumer goods (supply chain and inventory management), turn on the light switch (load management), receive a package delivery (vehicle routing), or fill up the car (oil refinery management). People seem to understand instinctively what biology, chemistry, or astronomy is. But since no one can understand what OR is, the idea of OR as a field does not propagate through society. As a result there is limited research funding for OR, few students choose to study OR as such, too few companies call on OR specialists for help—and to my knowledge, no OR person has ever appeared in a movie or TV show. In our age, what fails to appear in the media does not exist. As for CP, its obscurity could hitherto be attributed to its youth, but it is now moving into adulthood and may face the same public relations problems as OR. Consider the fact that, even today, many if not most OR people have little idea what CP is. Given this, what chance does CP have for recognition among a wider public?

Identifying a field with its techniques is intellectually as well as practically unsatisfying. Research is driven by curiosity, a desire to understand some phenomenon. At the root of every discipline lies a mystery: the mystery of life, matter, the stars, number, or space. It is the mystery that fascinates us and impels us to know more. We develop whatever techniques we need along the way. It is this urge to understand that energizes a field and gives it longevity.

8. The Task of CP/OR

CP and OR do not address ancient and enduring questions as do physics, biology, and cosmology. Yet as human activities take on global proportions and mind-boggling complexity, it becomes increasingly urgent that we understand how to coordinate them. Our lives are enmeshed in vast webs of activity that provide us food, clothing, transport, waste disposal, security, health care, information, financial services, and justice. No one fully comprehends

2. The community often finesses the issue by calling itself OR/MS. When the Operations Research Society of America (ORSA) merged with the Institute for Management Sciences (TIMS) in 1995, there was considerable debate about what to call the new organization, since the name would have implications for the definition of the field. The solution was to mention both: Institute for Operations Research and the Management Sciences (INFORMS).

these systems, and somehow they work to a greater or lesser degree. Yet we can no longer afford to let them evolve as they please. We are running out of raw materials, space, energy, and disposal options. Our social and economic systems are unstable, and just distribution is a very distant goal. We must increasingly manage our activities, or design our systems to work efficiently with less management.

Good management and efficient systems require resource allocation, product design, production planning, inventory management, network design, distribution, and scheduling—in other words, problems addressed by CP/OR. We cannot solve these problems unless we can wrap our minds around them. We must have theories that tell us what is possible and how to achieve it. These theories do not answer the perennial questions, but they address some of the most pressing questions of our age. We must keep our eye on these questions and shape our two disciplines accordingly.

I just spoke of disciplines in the plural, but if CP and OR have the same theoretical and practical task, then there is nothing to distinguish them—except their techniques, which of course should be no distinction, since a field should not be defined by its techniques. Does this mean CP and OR are the same? It does. This is why I do both and hope that, at some point, they will be subsumed by a unified field that investigates the structural properties of human activity.

9. Practical Implications

The good future sketched above for CP/OR requires some changes in professional practice. First, since CP/OR is to be an empirical science, the main criterion for journal publication should be the explanatory value of the model rather than the speed of the algorithm. The model should be empirically valid, in the sense that it mirrors the combinatorial structure of a real activity. An interesting paper may be publishable before its model is confirmed, as sometimes occurs in other empirical sciences (the general theory of relativity was published before critical confirming measurements were made during solar eclipses (Einstein, 1916)). But verisimilitude is an ultimate goal in both physics and CP/OR.

This does not mean that a model must prove itself economically useful before it is worthy of publication. Real-world verification is the ultimate goal for purely intellectual reasons. In fact, a model should be publishable even if it does not pay off, so long as it is enlightening. Even the empirical refutation of a model may be publishable. It is important to know that a plausible model fails to describe reality, and such findings are routinely published in other sciences.

Perhaps the most radical proposal in this future, to many readers, is the secondary role of computation. Computational speedups should not be a condition of publication, even if this departs from widespread practice in both CP and OR today. Although computation should remain a more urgent concern in CP/OR than in many sciences, much as it is in fluid dynamics, it should serve a larger goal. Papers that enlighten us with realistic models should receive attention even when the computational problem is yet unsolved. Readers can rest assured, however, that there is much continuity with current practice in the proposed future. A mathematical style—theorems, proofs and the like—remains central in CP/OR, because the field aims at structural analysis, and computational methods continue to exploit mathematical structure.

Since the primary aim of CP/OR is to create explanatory models, current modeling practices should change, particularly those of OR. Long lists of inequality constraints should be replaced with a limited number of metaconstraints, akin to global constraints, that reveal the structure of the problem. The practice of modeling with global constraints should be further refined, not primarily to serve the interests of computation, but to elucidate the structure of the problem. The computational benefits will follow, as they normally do when one exploits problem structure.

We should be careful to distinguish problem statements from models. This is particularly an issue in OR, which builds problem libraries like MIPLIB, a collection of mixed integer programming (MIP) instances. MIPLIB reflects the thinking that problems should be written in a standard form that can be fed into any solver. Yet this thinking confuses problem and model, and it illustrates how far OR has strayed from a modeling focus. Although MIPLIB is viewed as a library of problem instances, it is actually a library of *models* (MIP models). In many cases, the problem that gave rise to the MIP model is not even recorded. Often no citation to the literature is given, and it may be next to impossible to guess the original problem from the model. This precludes the possibility of writing a model that makes the problem easier to understand or solve. The OR community would see this as a serious drawback if it were centrally concerned with modeling. A problem library more appropriate to the modeling task would distinguish the problem from the model. It would look more like CSPLIB, which describes each problem in words, provides instances in a format that suits the problem, and occasionally suggests one or more ways to model it. Perhaps CSPLIB has this form out of necessity rather than virtue—due to CP's lack of a standard formulation rather than its focus on modeling—but it nonetheless sets a good example.

Our software tools should reflect the centrality of modeling. Except in systems designed for particular problem domains, models are now primarily built by writing constraints in a formal language that is hard to read and unforgiving of syntax errors. We should take full advantage of the graphical user interface to empower modelers. A metaconstraint should be invoked by opening a window on the computer screen, not by typing a statement. The window should present various options for refining the constraint and importing data. The model as a whole should be depicted graphically, with an opportunity to click on modules for a more detailed look. As the model is built, the software should repeatedly solve it and provide postoptimality analysis (much as a compiler provides diagnostics) so that the modeler can track how closely the model captures the phenomenon of interest. Postoptimality analysis in general should become a major focus. The system should never report a single solution in isolation, as is routinely done today.

Commercial systems currently provide some of these features, because practitioners want them. But they are seen as frills to attract customers rather than as central to the mission of CP/OR. The research community should take them no less seriously than solution methods. Companies should put modeling specialists as well as algorithmic specialists on their development teams.

Since modeling is the master and computation the servant, no computational method should presume to have its own solver. This means there should be no CP solvers, no MIP solvers, and no SAT solvers. All of these techniques should be available in a single system to solve the model at hand. They should seamlessly combine to exploit problem

structure. Exact methods should evolve gracefully into inexact and heuristic methods as the problem scales up. Fortunately, recent research into integrated methods, reported in CPAIOR conferences and elsewhere, has taken us a good distance in this direction.

A good future for CP/OR is above all a change in orientation. Research papers will explore the mathematical structure of ever more human activities, rather than propose ever more domain filters or cutting planes. A richer variety of problems will appear in the literature. They will suggest new mathematical structures that lead to interesting, new concepts and theorems. Commercial software will provide powerful modeling tools. Someone may even make a movie about a sexy, young scientist who does CP/OR.

References

R. L. Ackoff. The future of operational research is past. *Journal of the Operational Research Society*, 30:93–104, 1979.

P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer, 2001.

M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows*. Wiley, 2004.

J. Carlier. One machine problem. *European Journal of Operational Research*, 11:42–47, 1982.

J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35:164–176, 1989.

C. W. Churchman. Operations research as a profession. *Management Science*, 17:B37–B53, 1970.

C. J. Corbett and L. N. van Wassenhove. The natural drift: What happened to operations research? *Operations Research*, 41(1):625–640, 1993.

G. B. Dantzig. Linear programming: The story about how it began. In A. H. G. Rinnooy Kan J. K. Lenstra and A. Schrijver, editors, *History of Mathematical Programming: A Collection of Personal Remembrances*, pages 19–31. Elsevier, 1991.

A. Einstein. Die Grundlagen der allgemeinen Relativitätstheorie. *Annalen der Physik*, 49: 769–822, 1916.

S. Gass. A perspective on the future of operations research. *Operations Research*, 35: 320–321, 1987.

A. M. Goeffrion. The purpose of mathematical programming is insight, not numbers. *Interfaces*, 7:81–92, 1976.

P. Horner. Top 10 reasons why management science belongs in business schools. *OR/MS Today*, August, 2003.

J. E. Kelley. Critical-path planning and scheduling: Mathematical basis. *Operations Research*, 9:296–320, 1961.

D. A. Samuelson. Figures from U.S. Bureau of Labor Statistics present mixed bag for OR/MS employment. *OR/MS Today*, April, 1996.

M. Trick. Raising social capital. *OR/MS Today*, April, 2002.