Interfaces
Vol. 38, No. 5, September–October 2008, pp. 356–366
ISSN 0092-2102 | EISSN 1526-551X | 08 | 3805 | 0356

informs

DOI 10.1287/inte.1080.0382

Optimization Tradecraft: Hard-Won Insights from Real-World Decision Support

Gerald G. Brown, Richard E. Rosenthal
Department of Operations Research, Naval Postgraduate School, Monterey, California 93943,
gbrown@nps.edu

This paper honors the memory of deceased coauthor Richard E. Rosenthal

Practitioners of optimization-based decision support advise commerce and government on how to coordinate the activities of millions of people who employ assets worth trillions of dollars. The contributions of these practitioners substantially improve planning methods that benefit our security and welfare. The success of real-world optimization applications depends on a few trade secrets that are essential, but that rarely, if at all, appear in textbooks. This paper summarizes a set of these secrets and uses examples to discuss each.

"Thou shalt never get such a secret from me but by a parable."
Shakespeare, *The Two Gentlemen of Verona*

Key words: decision-support systems; integer programming applications; linear programming applications.
History: This paper was refereed.

Clients consult specialists because they have real-world problems to be solved. Clarifying a problem statement by talking with a client or, better, getting first-hand experience with the client organization is very different from reading a textbook case study. (However, some clients might feel that your success would threaten their jobs.) In this paper, we offer advice that we learned from completing hundreds of optimization-based decision-support engagements over several decades. These are hard-won lessons based on field experience. As a practitioner of our optimization art, you must obtain some experience beyond textbook coursework before these suggestions will make complete sense to you. Thus, you will not find this material highlighted in any textbook. Providing decision support in the real world is difficult because it requires that you deal with enterprise data systems, legacy procedures, and human beings who might not share your passion for making things better.

We receive many phone calls from colleagues and ex-students who are working with optimization. Sadly, too many of these callers do not extol the wonders of optimization; rather, they lament practitioner problems in getting things to work right. Unfortunately, this may have given us a distorted view of the issues we address here.

In this paper, we present our tradecraft in the topical categories that we have used to collect our lessons learned. Even if you are not a practicing optimizer, we suspect you will find insights here.

Design Before You Build

We have had an astonishing number of opportunities to address problems with optimization models that have been implemented, but are behaving badly (e.g., they are very hard to solve, too large to solve, or produce strange results) and are not documented. They have been built without a design!

Documentation must—not should, must—include these three critical components:

- A nonmathematical executive summary,
- A mathematical formulation, and
- A verbal description of the formulation (Figure 1).

A *nonmathematical executive summary* must answer the following five questions, preferably in this order (Brown 2004a):

- What is the problem?
- Why is this problem important?

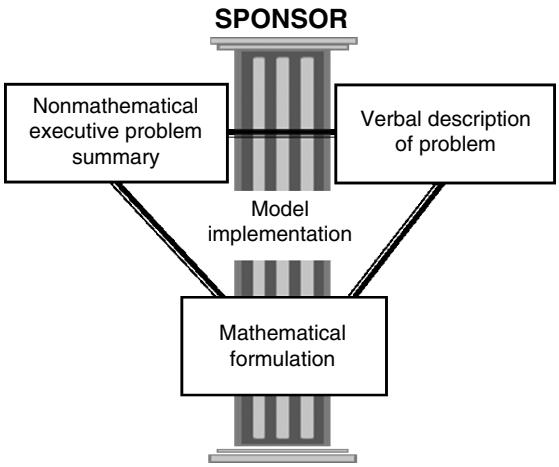

Figure 1: The model sponsor will only likely see the nonmathematical executive problem summary and the verbal problem description. The actual model implementation must be embedded with these two essential documents and with a mathematical formulation. In our experience, there is no substitute for any of these components.

- How would the problem be solved if you were not involved?
- What are you doing to solve this problem?
- How will we know when you have succeeded?

Express your executive summary in your executive sponsor's language, rather than in technical jargon. If you have trouble writing such a summary in less than five pages, you are not ready to proceed. The following tricks will make writing your summary easier and more effective:

- Have a nonanalyst read your executive summary to you, out loud,
- Ask this reader to explain your executive summary to you,
- Listen well, and
- Revise and repeat.

A *mathematical formulation* should include the following in this order (Brown and Dell 2007):

- Index use (define problem dimensions),
- Given data (and units),
- Decision variables (and units),
- Objectives and constraints, and
- (perhaps) a dual formulation.

Remember to define terms before using them. The earliest definition of such a standard formulation format appears in Beale et al. (1974). To distinguish inputs from outputs, adopt a convention such as using lowercase for indexes and data, and uppercase for decision variables.

A *verbal description of the formulation* (Figure 1) explains, in plain English and in your executive sponsor's language, what each decision variable, objective, and constraint adds to the mathematical model. It gives you the opportunity to define what the mathematics means and why each feature appears in your model. Avoid literally translating mathematics into English. For example, avoid saying "the sum of X over item subscripts i must be no more than m for each time subscript t." Instead, say "the total production of all items must not consume more raw material than will be available in any year." Do state and justify any simplifying assumptions (some examples include "our planning time fidelity is monthly, with a 10-year planning horizon," and "we allow fractional production quantities of these large volumes").

Bound All Decisions

Bounds restrict the domain of every decision. An unbounded variable does not exist in our real, OR analyst's world. Establishing bounds for each decision variable is a trivial concept that is often ignored. While any reasonable optimization solver will do this automatically, the solver cannot tell you that its analysis is based on bogus data or missing features in your model. If you manually apply simple ratio tests (e.g., "If I had all the steel the world produced this year, how many automobiles could I build?") and get ridiculous answers (e.g., "2.1 autos," or "10 trillion autos"), you have discovered an error either in the data or in the description of the manner in which automobile production consumes steel. These conversions reveal an erroneous steel consumption rate per auto or a constraint that has no influence on your model; thus, you can jettison them.

Do you remember all the formal "neighborhood" assumptions that underlie your optimization method? Taylor's theorem makes any continuous function appear linear if you bound your decision neighborhood tightly enough. All your costs and technology likely exhibit nonlinear effects across widely varying magnitudes; however, they might not exhibit the same effects over a small neighborhood—the domain for which you are planning.

It is easier to branch-and-bound enumerate models with integer variables if the bounds on the integer variables are as tight as possible. This is worth addressing before you try to solve large models. If the tightest bounds that you can state permit a "large" integer domain, relax the integrality requirement and round the continuous result to the nearest integer. The inaccuracy that rounding inflicts will be no worse than one divided by the final value of the variable.

Bounding all your decision variables pays an unexpected bonus. Pull out your favorite optimization textbook and look at the basic theorems that might have seemed so hard in class. Notice how much mathematical lawyering becomes superfluous when you rule out the unbounded case. Voila!

Expect Any Constraint to Become an Objective, and Vice Versa

Important planning models almost always exhibit multiple, conflicting objectives. Get familiar with a "weighted average objective," and what it really means. Learn about "hierarchical (i.e., lexicographic) objectives," and how to coerce off-the-shelf optimization software into following your hierarchy. For example, you might maximize the highest-priority objective, and then add a constraint on this objective to maintain this performance in all subsequent solutions. Repeat this process with each lower-priority objective until these successive restrictions have addressed your entire hierarchy, or your model is so overconstrained that further restriction would be pointless. Using some algebraic modeling languages, you can automate all of this as a single model excursion.

You can see that there is a continuum (sic) between weighted objectives in a single monolithic model, and strictly hierarchical ones in a sequence of successive restrictions. It is possible to force hierarchical results by using wide-ranging values for weights; however, you might regret the attempt. Take care to use your model-generation logic to control a hierarchal-solution sequence, rather than try to force your optimization model to make this asymptotic transition from finite weights to the infinite weights required to render absolutely lexicographic results. Floating-point numerical errors increase in direct proportion to the relative magnitude of the terms in your additive weighted objective. You might be able to express such an objective; however, your solver will not see what you intend to be lower in the objective hierarchy.

In one of our engagements, we dealt with an extreme case with 14 objectives, each weighted at least an order of magnitude more than its predecessor in the weighted hierarchy. This was not a pretty numerical experience for the solver.

Expert guidance from senior executives frequently filters down to modelers as constraints (i.e., orders). In our experience, constraints deriving from literal interpretation of such guidance inevitably lead to an infeasible planning model. Discovering what can be done changes your concept of what should be done. This leads you to "aspiration constraints," a situation in which you determine how much of something you can maximize in isolation; you can then write a constraint saying, for example, "I'll settle for 90 percent of this isolated maximum." If you work with your senior sponsors using these simple methods, you will be able to guide them to give you better advice. As OR analysts, we may think that our job is to give advice; however, our real objective is to help our sponsors to make the right decisions.

Much of the relevant literature advises us on how to deal with multiple objectives. It does a nice job of defining and explaining concepts, such as pareto-optimality. However, simple ideas usually work best.

Classical Sensitivity Analysis Is Bunk—Parametric Analysis Is Not

Blind application of dual values, right-side ranging, and other textbook tricks offer little useful advice on how the solution will respond as the inputs *all* change. Even for the few models that are continuously linear, classical textbook sensitivity analysis is rarely useful. Some of the best off-the-shelf mathematical modeling languages and solvers do not support such analysis. We professors love to teach this "stuff." We will continue to teach it because it conveys lessons on the foundations of our optimization methods, and on how to interpret the quantitative (how much to do) and qualitative (what to do) influence of restrictions and relaxations.

However, in the real world, *plan on solving many model excursions*; do not hesitate to try this approach

because "it may take weeks to complete." In the past 15 years, improvements in linear program (LP) solvers and, in particular, in integer linear program (ILP, aka MIP) solvers and their controls have improved performance by a factor of at least 10,000 *independent of the much faster speeds of newer computers*. Some in our profession, especially the senior, experienced professors and textbook authors, still recall overnight batch processing of mathematical programming system (MPS) tapes. This is not a fond memory; therefore, our advice is simple—"get over it." All you need today is a reasonably endowed desktop or laptop computer. For almost any modeling engagement, we can expect to set up an optimization model that allows us to express a question and get an answer while our sponsor still remembers the question.

Model and Plan Robustly

Ensure that your model considers alternative future scenarios and renders a robust solution. There are many ways to capture this in your model; all boil down to arriving at a single plan that, if applied to any of your scenarios, solves that scenario with acceptable quality, and which you can express as some combination of feasibility and optimality.

In the military, we plan for what is possible, not what is likely; therefore, we seldom employ random variables to represent the likelihood of each alternative future. We use simulation to make quantitative (perhaps random) changes to data elements; however, we rarely randomly sample qualitative future changes. Senior planners use judgment to arrive at what they think is a fully representative set of deterministic scenarios. While there could be many theater-war plans, we normally only have one chance per year to request what we need to prepare for all of them.

We pay attention to the current defense-planning guidance. As we develop our model, we try to address the sponsor requirements. For example, suppose that our guidance is to fight and win one engagement while suppressing another, and then to fight the other and win it. If we do not have the option of selecting our favorites of 20 available war plans for such potential engagements, we might have to plan for 20-times-19 permutations of engagement pairs.

You might not be able to develop a plan that addresses all scenarios; thus, you could be motivated to search for a worst-case plan, which will distort your results. It is better to convey truthful insights to your sponsor than to delude yourself with baseless optimism. From the full scenario set, we can devolve to, for example, meeting a maximum subset of scenario requirements, or maximizing some gauge of scenario fulfillment. Whatever plan you select, do your best to document with exquisite clarity your assumptions and compromises that differ from the overarching defense-planning guidance. Despite apocryphal tales of the demise of analysts bearing bad news, an OR analyst who uses diplomatic, unambiguous language and careful analysis to deliver bad news will be a hero.

We seek the worst case among a reasonable set of outcomes that we control because that is what we are obligated to worry about and defend against. There are many commercial analogs to this advice. We find little to distinguish private-sector competition from military planning.

Model Persistence

Optimization has a well-earned reputation for amplifying small changes in inputs into breathtaking changes in advice.

Decision-support engagements typically require many model excursions, followed by analysis, followed by revisions and more model runs. When we have invested heavily in analyzing a legacy scenario, and must make some trivial adjustment to attend to some minor planning flaw, the last thing we want is a revision that advises major changes. This is always an issue with rolling-horizon models; it also arises when you make iterative refinements to a static model.

If your model is unaware of its own prior advice, it is ignorant. You can expect annoying turbulence and disruption when solving any revision of a legacy model. Any prescriptive model that suggests a plan, and, if used again, is ignorant of its own prior advice, is free to advise something completely, needlessly different. This will surely cost you the faith of your sponsor. Sometimes, there are many nearly optimal policies; however, if you have already promulgated one of these, it is now a legacy-planning standard that is worth trying to preserve.

Persistence means "continuing steadily in some course of action." This is exactly what we do with long-term optimization-based decision-support engagements. We must successively meld our sponsor's expert judgment with our model's optimal advice.

It is easy to add model features that limit needless revisions. To do this, you need to *make a published legacy solution a required input*, and then add model features to retain attractive features or limit needless revisions of this legacy. These persistent features might include the following (Brown et al. 1997):

- Do not change this legacy resource consumption by more than 2 percent,
- Between this legacy solution and any revision, add (or delete or change) no more than three of the binary options in this set,
- Do not change X unless you also change Y.

We give our students a handout showing them how to state integer linear constraints that express the ubiquitous logical relationships required in decision support (for example, for binary options A and B, "A only if B," "A and B, or neither," "A or B, but not both," or "A or B, or both"). We also show them how to state persistent guidance for revisions (because this information rarely appears in textbooks). For example, the Hamming distance between a legacy vector of binary decisions and a revision counts only the bit-wise number of changes. To solve a sequence of revisions, you can use constraints either to limit the number of revisions; in cases in which you are looking for a set of alternative courses of action to present to your sponsor for subjective evaluation, you can force diversity of each revision from *any* legacy solution (Brown and Dell 2007).

The literature suggests widely that in a facility location, for example, one should use a binary variable to represent each close-open decision with a fixed cost inflicted when we choose open. *We rarely get to apply this in the real world* because each facility might be in one of several states (e.g., open, open but idle, mothballed, closed, or disposed); the real problem is to decide which state transitions are best for the client. In even the simplest case, we have preexisting legacy facilities and their states and we choose revisions of those states; in these revisions, each before-after state pair has its own distinct, fixed transition cost. Multiperiod planning requires a binary variable for each state transition and a constraint to force choice of only one transition per decision.

Solution cascades (Brown et al. 1987, p. 341) solve a window of active constraints and variables moved over, e.g., time, fixing the values of each variable as the value determined when it was last in a window, for several reasons. For example, omniscient long-term optimization models sometimes are too clever about anticipating the distant future; we prefer more realistic time-myopic planning. We can also use persistent cascades to incrementally revise a plan locally while preserving its overall scheme. Sometimes, the cascade subproblems are much easier and faster to solve in large numbers than the seminal, monolithic model.

We also wonder why our literature pays scant attention to end effects. When we plan on using periodic state reviews over a finite number of planning periods, how do we plan to leave our system at the end of this planning horizon? There may be industry rules of thumb or policies on the admissible state of your enterprise (e.g., always have sufficient supply on hand to satisfy the next 90 days of demand). Lacking such guidance, we often plan further into the future than the planning horizon requires because we want to get some realistic representation of the actions up to exactly the end of the planning horizon (and discarding the further future results) (Brown et al. 2004).

Pay Attention to Your Dual

A conventional linear program equality constraint has an unrestricted dual variable that we can interpret as "this is how much it would be worth to relax this constraint by one unit."

An *elastic* linear program equality constraint uses a linear penalty per unit of violation below (or above) its minimal (or maximal) range. Allowing this constraint to be violated below (or above) either range at some finite penalty cost-per-unit violation bounds its dual variable (i.e., "this is the most it is worth to me to satisfy this constraint; otherwise I'll violate it, pay this penalty, and deal with the consequences"). There is no such thing as an infinitely valuable constraint.

Decision makers get paid to deal with infeasibilities and cannot rule them out in the real world.

When you convince your sponsor to work with you to state each constraint with a well-planned penalty for its violation, you have enormously enhanced your control and understanding of your decision-support model. Remember that a phone call beats a clever planning method every time. That phone call could be between you and your sponsor, or between the sponsor and a supplier, superior, or even the IRS. A written problem description or model statement could never have the level of impact that relaxing exasperating restrictions does. Managers are paid to make these calls and deal with infeasibilities.

Elastic constraints provide another surprise bonus: integer linear programming is much easier to deal with when you know a priori that every candidate integer solution in an enumeration is, by definition, admissible (i.e., satisfies the constraints, albeit perhaps with some penalties). In addition, if you set your elastic penalties carefully, you will be rewarded with remarkable improvements in linear-integer solution quality and solver responsiveness.

If you have a linear program, or can relax to one, state its dual. If you cannot write an abstract of the meaning of this dual, if you cannot interpret your dual at all, or if your dual is nonsense (e.g., unbounded or infeasible), your primal problem is ridiculous. OK, this is strong language. Amend this to read "your primal problem needs more attention before you are ready to use it."

Consider this example of a simple maximum-flow model that we have used for military planning and, since 9/11, for planning homeland defense. It includes a source node, a destination node, and a capacitated, directed network through which we wish to push the maximum-flow volume from source to destination. Write this primal linear program and solve it. Now, recover the dual solution. Admire these dual values and note that each arc on a minimum cut is distinguished by two incident dual values that differ. *If you want to attack this maximum-flow network and can cut these arcs, you have decapitated it*.

Interpreting linear programming duals is the foundation of decomposition (Brown et al. 1987) and the bilevel defender-attacker or attacker-defender models (Brown et al. 2006).

Spreadsheets (and Algebraic Modeling Languages) Are Easy, Addictive, and Limiting

OK, we have a new problem; we need a quick answer; we need database support for model development and cataloging solutions; and we need a graphical user interface that supports ad-hoc analysis and graphical output. Thus, we must either spend a long time and a small fortune developing a purpose-built graphical user interface or use our off-the-shelf office software suite.

Spreadsheets with embedded optimization solvers are inviting. Even executive sponsors likely know how to bring up a spreadsheet; therefore, you will gain immediate acceptance by adopting this familiar "look and feel" standard. In addition, you will be able to catalog and display a spreadsheet solution immediately by using the tools you use in your integrated office software suite daily.

However, spreadsheets support only two-dimensional views (and pivot tables) of many-dimensional models; they exhibit "dimensional arthritis"—they can support a many-dimensional model; however, they do not do it easily or naturally (Geoffrion 1997).

We get many calls from spreadsheet users who wonder why their optimization results either take forever or generate incorrect results. One of the first questions that we ask is, "how much did you pay for the solver you used?" Consider spending a few thousand dollars per seat on a well-known, off-the-shelf, supported, and documented, commercial-quality optimization package. In addition, before you commit to using any solver, check the credentials of the optimization software provider and verify how you will get help if you have problems.

Modeling languages are crafted to accommodate multidimensional models; they feature interface links to all contemporary database, spreadsheet, and presentation managers, and make great prototypes. However, even if a prototype works and gains acceptance, the modeling language used for prototypic implementation might not make a good decision-support tool. Some modeling languages isolate models from off-the-shelf commercial solvers. They do not provide good support for large-scale, indirect-solution methods (for example, column generation or

decomposition). If you are working on an important problem, why would you jettison 40 years of experience in solving it well, and, instead, simplify and aggregate away essential details merely to be able to mechanically generate and solve problem instances?

The transition from hasty prototype to production-model generator and interface is not easy. However, in our experience, the results always justify the investment. The use of a commercial-quality optimization package could reduce your model-generation and solution times from hours to just seconds (Brown and Washburn 2007).

Heuristics Can Be Hazardous

A heuristic—whether a simple rule of thumb or a well-known local search method—is so easy to explain and implement that we are often tempted to use one in lieu of more formal methods. Heuristics might not require optimization software and might offer a tantalizing first choice to quickly assess a "common sense" solution. However, heuristics should rarely be your first (or only) choice. Geoffrion and Van Roy (1979) offer some simple, exquisite examples that they have used with executives to show how blind adoption of common-sense heuristics can bring you grief.

We can also develop bounds on the best solution possible, although this is not as much fun to do as building a solution-seeking method. Without some similar bound, our advice is of unknown quality. This quality certification is important: *a bound on the value of the best possible solution is just as important as the best solution you have*.

A mathematical optimization model takes longer than a heuristic to develop, and perhaps to solve; however, it can provide a bound. We develop models of relaxations of very hard problems merely to recover the bounds that they provide. Lacking a trustworthy assessment of the quality of your advice, you are betting your reputation that nobody else is more scrupulous or just plain luckier than you are.

While publishing a bound with your solution is the right thing to do, there is a risk. We have been told: "Hey, you're leaving money on the table!" Well, maybe we are and maybe we are not. At least, we are honest about the possibility.

The *interval of uncertainty* is what we call the interval between the value of a solution and a bound on the value of the best-possible solution (various sources exist, including integrality gap, decomposition gap, Lagrangean gap, and duality gap). When you compare two alternative scenarios, you can be absolutely sure about the winner if the two intervals of uncertainty are disjoint, no matter how large each of these intervals is. Realizing this, you can work only hard enough to find a distinguishing difference—and no harder.

We have also been in a private-sector competition in which our heuristic competitors wrote the sponsor and said, "these guys admit their solutions may not be right." Boy, they thought they got us there, didn't they? To this, they responded "but, our method gets better solutions the longer you run it." This reminds us of the difference between "known unknowns" and "unknown unknowns." We can work with the former; we get nightmares from the latter. While a heuristic might suggest a provably better plan than the plan the enterprise is using currently, you will never know how much more you might have discovered. Would we implement a solution with no quality assessment? No, thanks.

We have also been told (sigh, and have read in the literature) that "this ILP is NP-hard, so we use a heuristic." Please. Even if (ahem) you prove that your ILP is NP-hard (an essential reduction proof that is still absent from our literature too frequently), this only means it is as hard as many other problems that are routinely and reliably solved to good tolerance. How much better is a heuristic with polynomial run time than a bounded ILP enumeration, which benefits from hundreds of years of research and experience by our optimizers? In addition, is the heuristic really any faster?

The simplex method has been criticized for its exponential worst-case run time on polynomially complex linear programs. Given its excellent average performance on an immense diversity of real-world linear programs, the worst-case run time limit is a poor excuse to adopt an alternative solution method. We have a good idea of the classes of problems for which the simplex method works well.

We prefer to solve any model that we can, even approximately, using conventional mathematical optimization and the best software we have. If we

convince our client that our suggested planning tool is worthy, software that costs a few thousand dollars per seat should not be a problem.

In cases in which the cost per seat would be too high to distribute the best software we have, or the number of seats required is necessarily high, and the model admits a heuristic solution, we try to develop a heuristic. Using our best software, we test empirically to assess performance. If we distribute the heuristic, we maintain a backup with our more-expensive software to objectively assess any curious performance in the field. At the Naval Postgraduate School, this means that we must maintain computers and software at various classification levels in appropriately secured facilities. While this requires a significant investment in hardware and software, it is essential to providing a safety net for fielded heuristic solvers.

We have encountered other obstacles both in the government and in the private sector with "enterprise standard" computers that are not allowed to run "foreign" executables and "exotic" applications, such as our optimization models. For example, Navy Marine Corps Intranet (NMCI), which governs 351,000 computers, is the largest standardized internal computer network worldwide (Electronic Data Systems 2006). Presumably this standardization has had benefits for "one size fits all" IT support. However, it has been a continuing headache to us. We cannot afford to have each of our models "vetted" and "approved" (a process that takes many months and many thousands of dollars) for NMCI. Accordingly, we have developed heuristics that can run, for example, with Visual Basic within Microsoft Excel on a standard NMCI computer. We have also developed applications that run exclusively on a universal serial bus (USB) drive that can be connected to a NMCI computer.

We have also had to purchase computers, install our applications, and ship these to our clients. We refuse to confirm or deny where these clients serve, or if they also have their own private computers to do mission-essential work outside of NMCI. We do whatever is necessary to complete our missions.

Perversely, one of the most influential arguments for heuristics, and against excellent, off-the-shelf commercial optimization solvers, is the Draconian license managers of these solvers, which treat paying clients like criminals. We have seen many cases, in academe and in industry, where a good solver would have helped; however, it was rejected because of the sheer IT burden it would cause—that of struggling with optimization-provider sales persons, computer-specific, immobile license keys, and license-manager hassles.

Modeling Components

Models usually exhibit a variety of functional components that express different aspects of the modeled enterprise. Observe how this enterprise is organized and mimic this with your model. For example, when production plans influence financial plans, link these components with "passenger variables" (a passenger variable does not change the degrees of freedom in your model because it is defined by an equation) that isolate and highlight this communication between components. Choosing passenger variables deserves some care; you are trying to capture how the connected enterprise components communicate with each other.

You might think that cluttering your model with superfluous passenger variables and defining equations makes the resulting, larger model harder to solve. Fortunately, solvers employ "presolve" features that quickly identify "rank-one" algebraic redundancies (e.g., those that are identifiable without substituting more than one variable for its defining equation); remove them from the model before you solve it; then substitute them back in when you have completed the solution.

Incremental development of components offers an added benefit. During this phase of development, you need only work with representatives of the enterprise component that you are currently modeling; thus, you can focus without distraction on the lexicon, operation, fidelity, and key issues to capture. Better yet, you can arrange each component to be optimized in isolation during development and testing. Fix or constrain the passenger variables linking to other components, run the component alone, and unwind any mischief that appears in this localized exercise.

Designing Model Reports

Design model reports to match those that planners are already using.

It is not unusual to spend as much time in reporting as in modeling. For example, if you find that a Gantt chart is a key display that manual planners use, mimic it. If your model has significance for the enterprise, i.e., if your optimized plans can materially change profitability, plan on producing a set of operating statements. Such statements might contain a cash flow report, income statement, and balance sheet, including the most important gauge—return on owners' equity. This is difficult work because preparing such statements requires much enterprise operating data that you would not otherwise need. The payback for doing this foundation work is two-fold: you gain a deeper appreciation for where and how your model can influence the enterprise, and these synthetic reports will get the attention of your sponsor.

For example, if your advice might require raising significant amounts of funding (e.g., by borrowing, selling stock, issuing bonds, or diverting funds from other uses), the sources, methods, and forecast consequences of such fundraising are essential features of your model. If your objective is earnings per share, and both earnings and number of shares are discretionary, you have a ratio of decision variables that you might (or might not) be able to back out algebraically into a linear (sic) integer program. While this greatly complicates your modeling, it is essential to your reporting.

To our knowledge, the earliest example of such operating-statement reporting appears in paired papers by Bradley (1986) and Geoffrion (1986), who advised the board of directors of General Telephone and Electric (GTE) Corporation how to commit huge capital improvements with substantial impact on corporate results. Contributions by their GTE cohorts in this modeling project accompanied these papers. These authors generously provided us with all their historical client notes and model source code; we have dissected these and reapplied their methods.

We have had the distinct pleasure of working with both closely held companies and sole proprietorships. These owners quickly grasped optimization and its nuances, including integrality gaps, duality gaps, model fidelity, and uncertainty. Because their own money is at stake, they really engaged with the details and valued these operating statements. We have also had experience with scrupulously run, publicly held corporations; they also valued operating-statement outputs, but with not with the level of intensity of private entrepreneurs.

An added advantage accrues from reporting in terms of operating statements. The managers of various "stovepipes" (i.e., enterprise components that are strongly intraconnected, but weakly interconnected) in the enterprise can see their business component and its interaction with others. This provides a level playing field among these managers, and encourages them to plan, negotiate, and speak in a common language. We have seen cases where, for example, marketing wants to make its quarterly "numbers" for incentive bonuses, finance seeks goals that are stated in terms of float, accounts receivable currency, and cash-versus-debt positions, and manufacturing strives to meet production-standard goals. This is akin to the fable of blind men each touching one part of an elephant's anatomy, and guessing what the animal looks like. If you gather these managers in the same room and ask them to look at the same integrated operating reports, wondrous insights will follow.

Optimization also enables the generation of reports that management might not have known were possible. For example, it is easy to embellish a customary demand-fill rate report with an estimate of the total landed profit (or loss) accruing from those sales. Wow, this gets attention!

Design model outputs that are directly useable as model inputs. In practice, we frequently repeat model applications to iteratively revise our advice with small changes.

Conclusion

You may ask "why aren't these simple topics part of basic optimization course work?" We have been asked this before, and respond: "where were you when these pages were blank?" These ideas may be simple; however, we know of no other source of instructional materials that addresses these real-world concerns.

While many analysts have successfully applied optimization to real-world problems, few will admit the failures and false starts that too frequently delay a planning project. For example, INFORMS Edelman presentations include some very impressive results; understandably, however, they rarely discuss the failures that occur on the path to completion. You might

seek out these authors to learn, as we have, that the topics we report here are ubiquitous.

We have invested heavily to incorporate these principles into our graduate courses. In our program, each student is part of a group; the students attend a tightly coordinated, lengthy sequence of optimization core classes as a cohort. Thus, we have the luxury of getting to know and teach them individually and as a group over an extended period. While we have had some success in helping them to understand the material, it is not at a sufficiently high level. We have concluded that the only way students will appreciate the value of some of our advice, which might admittedly be tedious to implement, is through experience.

Accordingly, we try to convey these ideas to our military-officer students using both humorous, self-deprecating case studies of our past peccadilloes and homework exercises. However, we also realize that this will not make much of an impression until the student has had some seasoning. We include a continuing, evolving copy of this document in our course materials; we also give each graduate a "lifetime money-back guarantee" to call us later, admonishing them to have this document in hand when they do (Rosenthal 2007).

Suffice to say we have seen the same problems arise scores of times, even for very experienced operations researchers; we have cataloged some in this paper, along with our prescriptive cures.

We wish you the best of luck in helping us to extend our reach with prescriptive optimization-based decision support to make our world better and more secure.

Acknowledgments

This paper derives from decades of modeling engagements, many of which were exigent exercises assisting colleagues and past students. Successful rescue drills earned us a reputation, which led to an invited plenary tutorial at a Military Operations Research Society meeting (Brown 2002) and another at an INFORMS practice meeting (Brown and Rosenthal 2005). Along the way, we were asked to publish a "how to" guide for documenting optimization (Brown 2004a). Kirk Yost encouraged an intermediate "secrets to success" publication (Brown 2004b). In this paper, we focus on improving models that are *correctly* formulated. By collecting a rogues' gallery of examples that frequently lead to confusion (Brown and Dell 2007), Rob Dell helped us hone these topics and isolate the most common mistakes leading to *incorrect* formulations. We are aware that the references we cite here are insular. This is not an oversight. Our advice is so opinionated, we hesitate to implicate others. We also want to present a self-consistent, unified view of our complicated topic. These references are postcards home from a life journey in optimization. We credit our close colleague, Art Geoffrion, for his many insightful observations about the conduct of decision-support engagements (Geoffrion 1976a, b; Geoffrion and Van Roy 1979; Geoffrion and Powers 1980; and Geoffrion 1986, 1997). Most of all, we are grateful to so many students who have confronted real-world problems using the optimization tools we teach, and have claimed the "lifetime money-back guarantee" that we grant each of them to come back at us and complain that "neither my textbooks nor my notes from our courses explain this." *You students were right*. We fixed this with each of you and learned a lot along the way. We thank each of you. (And, every one of Distinguished Professor Rosenthal's many such warranties, public and personal, will be honored by me, and by my colleagues. Just get in touch with us.)

References

Beale, E. M. L., G. C. Breare, P. B. Tatham. 1974. The DOAE reinforcement and redeployment study: A case study in mathematical programming. P. I. Hammer, G. Zoutendijk, eds. *Mathematical Programming in Theory and Practice*. Elsevier, New York, 417–442.

Bradley, G. H. 1986. Optimization of capital portfolios. *Proc. National Comm. Forum* **86** 11–17.

Brown, G. G. 2002. Top ten secrets for successful application of optimization. *Military Oper. Res. Soc. Annual Meeting*, Ft. Leavenworth, KS, June 19.

Brown, G. G. 2004a. How to write about operations research. *PHALANX* **37**(3) 7 ff.

Brown, G. G. 2004b. Top ten secrets to success with optimization. *PHALANX* **37**(4) 12 ff.

Brown, G. G., R. F. Dell. 2007. Formulating linear and integer linear programs: A rogues' gallery. *INFORMS Trans. Ed.* **7**(2, January).

Brown, G. G., R. E. Rosenthal. 2005. Secrets of success with optimization. *INFORMS Practice Meeting*, Palm Springs, CA, April 18.

Brown, G. G., A. R. Washburn. 2007. The fast theater model (FATHM). *Military Oper. Res.* **12**(4) 33–45.

Brown, G. G., R. F. Dell, A. M. Newman. 2004. Optimizing military capital planning. *Interfaces* **34** 415–425.

Brown, G. G., R. F. Dell, R. K. Wood. 1997. Optimization and persistence. *Interfaces* **27** 15–37.

Brown, G. G., G. W. Graves, M. D. Honczarenko. 1987. Design and operation of a multicommodity production distribution system using primal goal decomposition. *Management Sci.* **33** 1469–1480.

Brown, G. G., G. W. Graves, D. Ronen. 1987. Scheduling ocean transportation of crude oil. *Management Sci.* **33** 335–346.

Brown, G. G., M. Carlyle, J. Salmerón, K. Wood. 2006. Defending critical infrastructure. *Interfaces* **36** 530–544.

Electronic Data Systems. 2006. EDS signs NMCI contract extension to 2010. Retrieved April 24, 2008, http://www.eds.com/news/news.aspx?news_id=2905.

Geoffrion, A. M. 1976a. The purpose of mathematical programming is insight, not numbers. *Interfaces* **7**(1, November) 81–92.

Geoffrion, A. M. 1976b. Better distribution planning with computer models. *Harvard Bus. Rev.* **54**(4, July–August) 92–99.

Geoffrion, A. M. 1986. Capital portfolio optimization: A managerial overview. *Proc. National Comm. Forum* **40**(1) 6–10.

Geoffrion, A. M. 1997. Maxims for modelers. Retrieved April 18, 2008, http://www.anderson.ucla.edu/faculty/art.geoffrion/home/docs/Gudmdlg2.htm.

Geoffrion, A. M., R. F. Powers. 1980. Facility location analysis is just the beginning. *Interfaces* **10**(2, April) 22–30.

Geoffrion, A. M., T. J. Van Roy. 1979. Caution: Common sense planning methods can be hazardous to your corporate health. *Sloan Management Rev.* **20**(4, summer) 31–42.

Rosenthal, R. E. 2007. It's more than a job or an adventure. *OR/MS Today* (August) 22–28.