Vol. 7, No. 2, January 2007, pp. 153–159
ISSN 1532-0545 | 07 | 0702 | 0153

INFORMS
Transactions on Education

DOI 10.1287/ited.7.2.153
© 2007 INFORMS

Formulating Integer Linear Programs: A Rogues' Gallery

Gerald G. Brown, Robert F. Dell
Operations Research Department, Naval Postgraduate School, Monterey, California 93943
{gbrown@nps.edu, Dell@nps.edu}

"Convincing yourself is easy,
persuading a colleague is harder,
but proving it to a computer is hardest of all!"
R. Hamming, ca 1985

The art of formulating linear and integer linear programs is, well, an art: It is hard to teach, and even harder to learn. To help demystify this art, we present a set of modeling building blocks that we call "formulettes." Each formulette consists of a short verbal description that must be expressed in terms of variables and constraints in a linear or integer linear program. These formulettes can better be discussed and analyzed in isolation from the much more complicated models they comprise. Not all models can be built from the formulettes we present. Rather, these are chosen because they are the most frequent sources of mistakes. We also present Naval Postgraduate School (NPS) format; a define-before-use formulation guide we have followed for decades to express a complete formulation.

1. Introduction

Formulating linear and integer linear programs is an acquired skill, and developing this skill requires a lot of practice. We present some simple "how-to" examples selected for their usefulness and their likelihood to confuse.

Textbooks are full of complete examples that show how entire word problems are formulated, rather than how the constituent components of these formulations are assembled, step-by-step, into a final monolith. Stevens and Palocsay (2004) present a nice summary of typical textbook guidance, review relevant research on cognitive psychology and word problem solving, and suggest a sequence of formulation steps. They report this has helped their introductory management science business students translate formulation "word problems" into linear programs.

We have learned that the best way to teach our graduate engineering students the art of formulation is to distill common building blocks—ubiquitous components drawn from many large, complex, real-world models—and present these in isolation. The idea is to show how to convert verbal specification of a single concept into a mathematical description, and vice versa, and how to verify that this translation retains its intended meaning.

We call our primitive examples "formulettes." We have compiled these from experience as consultants, and with our colleagues and military officer students who exhibit a wide range of preparation and experience. For instance, our coursework ranges from a one-quarter optimization survey course to a three-quarter graduate optimization sequence, with following electives. We have learned to pay close attention to the common mistakes we struggle with, and our collection of formulettes expresses the essence of each common point of failure or confusion.

Surprisingly, a short set of examples covers almost all common sources of confusion.

We present each formulette using an identifying index, a short verbal description of what we want to do (written in quotes), an example of a mathematical formulation that achieves the desired result, and a "take away" from the example explaining why it appears in this guide.

Why not use this as a self test?
Rather than just glancing though our guide, and missing all the fun and most of the insights, read each example's verbiage, and try your hand at formulating a response.
Then, compare your result with ours.

If your answer differs from ours, how do you explain this? If they agree, can you see why we include this example in our rogues' gallery?

2. Linear Programming Formulettes

Each linear programming formulette, L1–L6, represents a category from a large library of drills we make our students solve for homework and on examinations.

For each formulette, write linear constraints in terms of the non-negative, continuous decision variables $X1$, $X2$, and $X3$.

(L1) "For each unit of $X1$, there must be at least 5 units of $X2$."

$$5X1 \le X2$$

The most frequent wrong answer is $X1 \le 5X2$. We always suggest trying a sample numerical value for $X1$ and deriving what that implies for a lower bound on $X2$. E.g., $X1 = 3$ yields $X2 \ge 15$.

(L2) "A port can load either 11 $X1$'s per week, or 45 $X2$'s, or 30 $X3$'s. What combinations of $X1$, $X2$, and $X3$ can be loaded in 10 weeks?"

$$(1/11)X1 + (1/45)X2 + (1/30)X3 \le 10$$

The most frequent wrong answer is $11X1 + 45X2 + 30X3 \le 10$. A unit check helps here: 11 "$X1$'s per week" times "$X1$'s" yields "11$X1$'s squared per week." Another common answer is the restriction $(1/11)X11 + (1/45)X2 + (1/30)X3 = 10$ that eliminates feasible options such as loading nothing at all: $X1 = X2 = X3 = 0$.

(L3) "There must be exactly 7 units of $X1$ for every 9 units of $X2$."

$$9X1 = 7X2$$

This is called a fixed-assay (or fixed-recipe, or fixed-proportion) constraint. The formulette might be better expressed: $(1/7)X1 = (1/9)X2$. Here too, we suggest trying a numerical example for $X1$ and see what that implies for $X2$.

(L4) "Sorties of type $X1$ must constitute at most 33% of all sorties of type $X1$, $X2$, and $X3$."

$$0.67X1 \le 0.33X2 + 0.33X3$$

This is a discretionary recipe stating an extreme limit on admissible sortie mixes. We often don't simplify the expression as above, leaving it as $X1 \le (33/100) \cdot (X1 + X2 + X3)$. The confusion here usually arises because there is no fixed number of sorties given. Thus, we are stating a recipe that can be applied to any total number of sorties. A nonlinear expression here $(X1/(X1 + X2 + X3) \le 33/100)$ can also lend insight as an intermediate step before converting to a linear expression.

(L5) "What mixtures of punch from 80-proof $X1$, 100-proof $X2$, and 0-proof $X3$ are at least 30-proof?" (Proof is a number that is twice the percent by volume of alcohol present.)

$$(80 - 30)X1 + (100 - 30)X2 + (0 - 30)X3 \ge 0$$

This is another discretionary recipe that is easier to understand when written as: $80X1 + 100X2 + 0X3 \ge 30(X1 + X2 + X3)$. We point out equivalent forms using percent alcohol (divide by 2) and fraction alcohol (divide by 200). (A note from naval history: Burning a mixture of equal quantities of black gun powder and rum yields an ash that was a quick chemical "powder proof" of alcohol content easily performed at sea a few hundred years ago.)

(L6) "Process $X1$ produces 24.5 units of $X2$ and 73.1 units of $X3$ per hour."

$$X2 = 24.5X1$$

$$X3 = 73.1X1$$

The $X1$-process produces multiple outputs at a fixed, synchronous rate.

3. Integer Linear Programming Formulettes

Our first optimization course introduces binary decision variables right away, long before we teach how a solver accommodates such an embellishment. In the following examples, variables X are still continuous and non-negative, but variables Z are binary. The usual interpretation of a binary variable is that a value of one means "true," "yes," "select," or "do," while zero respectively means "false," "no," "reject," or "don't."

(B1) "You can launch satellite $Z1$ only if you have chosen a compatible booster $Z2$." "$Z1$ ONLY IF $Z2$" "$Z1$ is sufficient for $Z2$" "$Z1$ implies $Z2$"

		Z2	
		0	1
Z1	0	1	1
	1	0	1

$Z1 \le Z2$

At the left is the "truth table" for this expression. Our students are familiar with a truth table so we use one to systematically enumerate each case in the English description and/or show the constraint expresses the intended relationship.

(B2) "Z3 can be produced if and only if a machine Z1 and a worker Z2 are available." "Define Z3 as 'Z1 AND Z2'":

$$\begin{array}{cc|cc} Z3 = & & \multicolumn{2}{c}{Z2} \\ & & 0 & 1 \\ \hline Z1 & 0 & 0 & 0 \\ & 1 & 0 & 1 \end{array} \qquad \begin{array}{l} Z3 \le Z1 \\ Z3 \le Z2 \\ Z3 + 1 \ge Z1 + Z2 \end{array}$$

At the left is the truth table. A common wrong answer is the expression "$Z3 = Z1 * Z2$." The continuous relaxation of "$Z3 = Z1 * Z2$" is NOT linear.

Systems working synergistically often have increased capability that cannot be linearly expressed solely in terms of, for instance, Z1 and Z2. The binary variable Z3 can be used to convey any sub- or super-linear synergistic effect between Z1 and Z2. If the objective function maximizes the value of Z3, the last inequality is not needed.

(B3) "Project Z3 can be funded if and only if project Z1 or project Z2, or both projects are funded." "Define Z3 as the 'INCLUSIVE OR' of Z1 and Z2."

$$\begin{array}{cc|cc} Z3 = & & \multicolumn{2}{c}{Z2} \\ & & 0 & 1 \\ \hline Z1 & 0 & 0 & 1 \\ & 1 & 1 & 1 \end{array} \qquad \begin{array}{l} Z3 \le Z1 + Z2 \\ \\ Z3 \ge Z1 \\ Z3 \ge Z2 \end{array}$$

"INCLUSIVE OR" translates to "Z3 if and only if Z1, or Z2, or both Z1 and Z2." At the left is the truth table. If the objective function maximizes the value of Z3, the latter two inequalities are not needed.

(B4) "Packaging line Z3 can receive product from either processing line Z1 or processing line Z2." "Define Z3 as the 'EXCLUSIVE OR' of Z1 and Z2."

$$\begin{array}{cc|cc} Z3 = & & \multicolumn{2}{c}{Z2} \\ & & 0 & 1 \\ \hline Z1 & 0 & 0 & 1 \\ & 1 & 1 & 0 \end{array} \qquad \begin{array}{l} Z3 \le Z1 + Z2 \\ Z3 \ge +Z1 - Z2 \\ Z3 \ge -Z1 + Z2 \\ Z3 \le 2 - Z1 - Z2 \end{array}$$

"EXCLUSIVE OR" translates to "Z3 if Z1 or Z2, but not both Z1 and Z2." At the left is the truth table. Students usually don't find this formulette obvious at all. Do you?

(B5) "NOT Z."

$$1 - Z.$$

This "binary reflection" turns out to be useful, especially in conjunction with the previous logical conditions.

(B6) "From the set $Z1, Z2, \ldots, Zk$, select at most one."

$$Z1 + Z2 + \cdots + Zk \le 1$$

This is called a "pack" constraint.

(B7) "From the set $Z1, Z2, \ldots, Zk$, select exactly one."

$$Z1 + Z2 + \cdots + Zk = 1$$

This is called a "partition" constraint.

(B8) "From the set $Z1, Z2, \ldots, Zk$, select at least one."

$$Z1 + Z2 + \cdots + Zk \ge 1$$

This is called a "cover" constraint.

(B9) "From the set $Z1, Z2, \ldots, Zk$, select more than three."

$$Z1 + Z2 + \cdots + Zk \ge 4$$

This is called a "cardinality" constraint. Did you catch that "more than three" is at least four? While $Z1 + Z2 + \cdots + Zk > 3$ is a mathematically correct expression, its linear programming relaxation isn't tight. Our students know the divisibility assumption and that there is no practical difference between $>$ and $\ge$ in a linear program, or in an algebraic modeling language.

(B10) "If you build a warehouse $Z = 1$, you can store up to 13 tons of X in it." "If $Z = 0$, then $X = 0$, but if $Z = 1$, then $X \le 13$."

$$X \le 13Z$$

Z is called the controlling variable and X is the controlled variable. This constraint is called a "variable upper bound."

(B11) "If you build a warehouse $Z = 1$, it must be used to store at least 56 tons but no more than 141 tons." "If $Z = 0$, then $X = 0$, but if $Z = 1$, then $X \in [56, 141]$."

$$X \ge 56Z$$

$$X \le 141Z$$

X is called a "semi-continuous" variable, controlled by binary variable Z.

(B12) "Use binary variables $Z1, Z2, Z3$, and $Z4$ to restrict variable X to a discrete value in the range $5, 6, \ldots, 15$."

$$X = 5 + 1Z1 + 2Z2 + 4Z3 + 3Z4$$

This is a binary factorization that can also be written as $X = 5 + 1Z1 + 2Z2 + 4Z3 + 8Z4$ with the additional constraint $X \le 15$. Generally, if you can accommodate a general integer variable directly, this is preferred to a binary factorization. However, once in a while we need to restrict some continuous variable to a domain that is a combination of points and intervals, and there are situations where an all-binary model is easier to solve with help from pre-solve reductions not applicable to general integers.

4. Subscripted Examples

Converting to notation using subscripts, here are a few more formulettes:

(S1) "Given baseline solution $(z_1^*, z_2^*, \ldots, z_n^*)$, make sure revision $(Z_1, Z_2, \ldots, Z_n)$ differs in at least one position (or variable)."

$$\sum_{i \mid Z_i^*=0} Z_i + \sum_{i \mid Z_i^*=1} (1 - Z_i) \geq 1$$

The left-hand side of this expression computes the non-negative Hamming distance between the baseline and any revision (Brown et al. 1997). You can use this to limit the number of bits that differ, or to force a number of bits to differ. Here, we just want a binary revision distinct in at least one bit. The first summation counts the number of baseline zeroes that are revised to one, and the second summation counts ones revised to zero.

(S2) "During each period t in the planning horizon, fly no more than $limit_t$ sorties $X_{i,j,m,t}$ using aircraft '$i = FA18$' to drop munition '$m = SL72$' against all targets j."

$$\sum_j X_{FA18,j,SL72,t} \leq limit_t \quad \forall t$$

We remind our students to always ensure each index is under control in an expression, and there are only three, mutually exclusive, exhaustive ways to control an index: Either

a. Specify its value explicitly as is done above for "$i = FA18$," or

b. Sum over its value as is done above for the j index, or

c. Specify that the expression should be formed for specified members of index sets as is done above for index t.

(S3) "For each period $t = 1, 2, \ldots$, in the planning horizon, model periodic review of a continuous process, where the period t purchases, buy_t, and period t disposals, $dispose_t$ are aggregated by planning period and the inventory state X_t is reckoned at the ***end*** of each period t. Start with an initial inventory of X_t $(t = 0)$ units."

$$X_t = X_{t-1} + buy_t - dispose_t, \quad t = 1, 2, \ldots$$

Clearly, we can formulate an equivalent model substituting "begin" for "end." Ambiguity about the timing of state reviews is a frequent source of trouble.

The time-recursive expression above can be expressed in time-cumulant form:

$$X_t = X_0 + \sum_{\tau \leq t} (buy_\tau - dispose_\tau), \quad t = 1, 2, \ldots$$

(S4) "For each period $t = 1, 2, \ldots$, in the planning horizon, begin $(t = 0)$ with initial inventory of units that are c years old, $X_{t,c}$. Then account for period-by-period purchases, buy_t, and retirements, $dispose_{t,c}$, to determine active inventory at the end of each period t."

$$X_{t,c} = X_{t-1,c-1}|_{c>1} + buy_{t-1}|_{c=1} - dispose_{t,c}, \quad t = 1, 2, \ldots; \; c = 1, 2, \ldots$$

To keep track of the age of each unit, each cohort c must be maintained as a separate commodity. This terse example shows how age-limited inventory expands the size of the model. We place conditional term $(c > 1)$ and $(c = 1)$ on the $X_{t,c}$ and buy_t variables in lieu of writing a separate constraint set for initial purchases.

Curiously, although inventory models appear in every textbook, aged inventory examples like this rarely appear. Yet, accounting for age-influenced costs and effects is a business bookkeeping requirement.

(S5) "Each ship may spend two contiguous months in port dry dock, but the dry dock can accommodate at most four ships during any month…"

Let $Z_{s,t}$ be a binary variable with value one if ship s starts a two-month port dry dock at the beginning of month t (and ends at the beginning of month $t+2$), and zero otherwise.

$$\sum_s \sum_{t'=\max(1,t-2+1)}^{t} Z_{s,t'} \leq 4, \quad t = 1, 2, \ldots, T$$

The binary variable definition above is not typically the first choice for our students. More common is "let $Z_{s,t}$ be a binary variable with value one if ship s is in dry dock during month t, and zero otherwise." This simplifies the above constraint to

$$\sum_s Z_{s,t} \leq 4, \quad t = 1, 2, \ldots, T$$

but requires additional constraints

$$Z_{s,t} \leq Z_{s,t-1} + Z_{s,t+1} \quad \forall s, \; 1 < t < T$$
$$Z_{s,t} \leq Z_{s,T-1} \quad \forall s$$
$$Z_{s,t} + Z_{s,t-1} + Z_{s,t-2} \leq 2 \quad \forall s, \; t > 2$$

to ensure a ship stays in dry dock for consecutive months. This results in a model that often takes longer to solve.

(S6) "During each period $t = 1, 2, \ldots$, in the planning horizon, a production line can either be in an operating, mothballed, or closed mode. The production line is open at the start of the planning horizon. If the line is ever closed, it must stay closed for the remainder of the planning horizon."

Let $Z_{m,m',t}$ be a binary variable with value one if the production line transitions from mode m at the start of week $t-1$ to mode m' at the start of week t.

$$\sum_{m,t} Z_{m,\text{“closed,”}\,t} \le 1$$

$$Z_{\text{“open,” “open,” “0”}} = 1$$

$$\sum_{m,m'} Z_{m,m',t} = 1, \quad \forall t$$

The first pack inequality ensures that if the line closes, it stays closed, the second definition fixes the mode at the start of the planning horizon, and the last partition equation regulates period-to-period transitions.

(S7) "Ensure the production line transitions from mode m (to any mode m') at the start of week t only if it was operating in mode m during the previous week."

$$\sum_{m'} Z_{m,m',t} \le \sum_{m'} Z_{m',m,t-1}, \quad \forall m, t > 1$$

There is usually a different operating cost for each mode, and a transition cost between each successive pair of modes, and the binary variables $Z_{m,m',t}$ give us the fidelity we need to represent this. By restricting permitted pairs of the m and m' indices, we can express conditions such as: "once operating, continue operating," "once closed, stay closed," and "the line can operate in period t only if it was operating or mothballed in period $t-1$." Here, a cost $c_{m,m',t}$ associated with transition decision $Z_{m,m',t}$ can represent a setup or a changeover.

Textbook examples usually express a simple fixed cost associated with an open-close binary state variable for each planning period, and no period-to-period consequences—we seldom find this simplification useful in practice.

5. Formulating in NPS Format

We have developed and always use what we have immodestly come to call Naval Postgraduate School (NPS) format for all our formulations. Further, to gain our advice and support, we require our correspondents to use this format, and help them do so. NPS format defines model indices, problem data, decision variables, the objective function(s) and constraints expressed in these terms, and finally offers a discussion of the formulation that clarifies any potentially confusing terms.

This is a define-before-use format that guides clear exposition, and can be directly implemented in any well-designed algebraic modeling language. NPS format has evolved from decades of experience, but we also heartily recommend a rather obscure, seminal reference by Beale et al. (1974).

An NPS format template contains the following sections, with the given headings, in the given order:

Index (and Set) Use [Cardinality or ∼Cardinality]
Defines dimensions and the cardinality or estimated cardinality of each index, and each index-tuple. (When not all index-tuples exist, we carefully document these key details.) We forecast our use of indices by including all functions of indices and mappings among indices. Cardinalities help assess the size of the resulting model.

Data [Units]
Presents each exogenous data element and its units (using the already-defined indices). Data derived from these exogenous sources are defined here, rather than later. Separating given data from data derived from this is important to avoid confusion between exogenous and endogenous effects in a formulation.

Decision Variables [Units]
Presents each decision variable and its units (using the already-defined indices). (We use UPPERCASE variable names, and lowercase for data and indices. Such a convention helps a reader to quickly see whether a model is linear, and understand its formulation.)

Formulation [Dual Variables]
Using prior definitions, there should be no surprises in these expressions of objective function(s), constraints, and variable domains. When applicable, we name the dual variable for each constraint.

Discussion
This explains the intent of each formulation feature, associating model nomenclature with that of the seminal problem we want to solve. Here, we have a chance to associate mathematical model notation with the lexicon of our client decision maker.

Ideally, we should be able to use this verbal model description to reconstruct the formulation, and vice versa.

6. Composite Formulettes

Each following small example illustrates some amalgam of the features introduced above, and shows how to answer some question that we have encountered many times in practice.

(C1) "There are k mutually exclusive, exhaustive production campaign options for a new missile. Each campaign option is characterized by a qualifying minimum number of missiles, a maximum number of missiles, a fixed campaign adoption cost, and a unit cost per missile. Further…"

Index Use [∼Cardinality]

$p \in P$ production campaign options [∼10]

Given Data [Units]

$fixed\_cost_p$	fixed adoption cost for campaign option p[\$]
$unit\_cost_p$	unit cost per missile in campaign option p[\$/missile]
min_units_p, max_units_p	minimum, maximum number of missiles in campaign option p [missiles]

Decision Variables [Units]

$MISSILES_p$	number of missiles to produce using option p [missiles]
$ADOPT_p$	binary decision to adopt campaign option p [binary]

Formulation

$$\min_{MISSILES,\,ADOPT} \sum_p unit\_cost_p MISSILES_p + \sum_p fixed\_cost_p ADOPT_p \quad \text{(C11)}$$

$$\text{s.t.}\quad MISSILES_p \geq min\_units_p ADOPT_p \quad \forall p \quad \text{(C12)}$$

$$MISSILES_p \leq max\_units_p ADOPT_p \quad \forall p \quad \text{(C13)}$$

$$\sum_p ADOPT_p = 1 \quad \text{(C14)}$$

$$MISSILES_p \geq 0 \quad \forall p \quad \text{(C15)}$$

$$ADOPT_p \in \{0,1\} \quad \forall p \quad \text{(C16)}$$

Discussion

The objective (C11) expresses the total cost of any solution in terms of the decision variables. Each constraint (C12) forces a selected production campaign option quantity to at least its minimum volume, and each constraint (C13) limits a selected campaign option to no more than its maximum volume. Constraint (C14) forces selection of exactly one binary campaign option. Variable domains are defined by (C15) and (C16).

This formulation can express arbitrary alternate campaigns. The alternate ranges of campaign missile quantities can overlap, or there may be gaps in missile quantities that no campaign can produce. Alternate campaign costs can be used to represent efficiencies of scale, learning curve, diminishing returns to scale, or any combination of these. Notice that because the index domain is defined in advance, the expressions using this index need not include these details, and are thus simplified.

(C2) "Each year in the planning horizon has a budget allocation, but you can borrow 5% or save 10% of this allocation, paying a penalty for any cumulative deficit or surplus carried forward from one planning year to the next, seeking to eventually make cumulative spending equal cumulative budget. Further, ..."

Index Use [∼Cardinality]

$t \in T = \{2008, 2009, \ldots\}$ planning year (alias τ)[∼30]

Given Data [Units]

$budget_t$	annual budget allocation for planning year t[\$]
pen_under_t, pen_over_t	penalty per dollar for carrying forward a cumulative deficit, or a surplus at the end of year t[\$/\$]

Decision Variables [Units]

$SPEND_t$	amount of money to spend during planning year t[\$]
$UNDER_t$, $OVER_t$	cumulative surplus, or deficit at end of planning year t[\$]

Formulation [Dual Variables]

$$\min_{SPEND,\,UNDER,\,OVER} \sum_t pen\_under_t UNDER_t + \sum_t pen\_over_t OVER_t \quad \text{(C21)}$$

$$\text{s.t.}\quad \sum_{\tau \leq t} SPEND_t + UNDER_t - OVER_t = \sum_{\tau \leq t} budget_t \quad \forall t\ [\pi_t] \quad \text{(C22)}$$

$$0 \leq UNDER_t \leq 0.10 \sum_{\tau \leq t} budget_\tau \quad \forall t\ [\lambda_t] \quad \text{(C23)}$$

$$0 \leq OVER_t \leq 0.05 \sum_{\tau \leq t} budget_\tau \quad \forall t\ [\alpha_t] \quad \text{(C24)}$$

$$SPEND_t \geq 0,\ UNDER_t,\ OVER_t \geq 0 \quad \forall t \quad \text{(C25)}$$

Discussion

The objective function (C21) expresses the total penalty incurred from any saving or borrowing from year-to-year. Each constraint (C22) balances total spending through the end of each planning year with the cumulative budget through the end of that year, accounting for any surplus or deficit year by year. At the end of each year, cumulative surpluses or deficits are limited respectively by a constraint (C23) and (C24). (C25) simply precludes negative spending. A penalty of 2.0 translates into a 100% per annum charge on any saving or borrowing. Penalties indexed by year can be used to convert to present value, if needed. Constraints (C22) are called *elastic goals*, and the *cumulative elastic budget goals* shown here are the most useful tool we have for long-term capital planning, where it is key to know when and how much flexibility to ask for. The dual variables provide additional guidance on the value of altering the budget. For example, π_t (the value of restricting cumulative spending allowed by the end of year t) is in the range $-pen\_over_t \leq \pi_t \leq pen\_under_t$ when constraints (C23) and (C24) are nonbinding.

See, e.g., Brown et al. (2004) for more complete formulations and real-world examples.

7. Parting Comments

The best way we know to teach our students the art of formulating linear and integer linear programs is by giving them ***lots of practice***: We pepper them with confusing formulettes, and build more interesting models from these. Alternately, we dissect large, complex models, to isolate such components.

A class examination expressed in terms of a set of independent formulettes is much less likely to overwhelm a student than some large word problem, but the demonstration of mastery of concepts is equivalent.

We also expose our students to our clients for optimization models. It is quite instructive to let a client tell his version of a problem that the students supposedly have already formulated and solved. The subsequent exchange is insightful.

Perversely, formulating a textbook problem intended for instruction can be harder than dealing with a real problem: You can call your client to get clarification, negotiate details and assumptions, or try alternate ideas. The ability to verbally express a formulette back to a client in the client's own terms aids substantially in this negotiation.

Sadly, our literature still publishes many articles that feature astonishingly muddled formulations. The result is, at best, confusion, but it's often worse than this. Our recommended "NPS standard form" is minimally restrictive. Much more ambitious, formal formulation methods have been suggested (e.g., Geoffrion's structured modeling 1987). We admire this formalism, but our goal is much more modest: We just want to teach students how to reliably get the mathematics to agree with the verbiage, and avoid errors.

Few clients understand a mathematical formulation verbatim—that's why they engage our help—so we must translate our formulation into words they do understand. The ability to express a client's problem in clear, unambiguous *client* language is key. Brown (2004) presents a complementary guide that highlights how to organize and present such exposition.

The few examples presented here represent about 90 percent of all formulation errors we encounter.

NPS standard form helps us catch the rest.

Acknowledgments

We have spent about six professor-decades learning from our military officer students how to teach these modeling methods: We are grateful for their instruction.

References

Beale, E. M. L., G. C. Beare, P. B. Tatham. 1974. The DOAE reinforcement and redeployment study: A case study in mathematical programming. P. L. Hammer, G. Zoutendijk, eds. *Mathematical Programming Theory and Practice: Proceedings of the NATO Advanced Study Institute, Figueira da Foz, Portugal*, June 12-23, 1972, Elsevier, New York, 417–442.

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

Brown, G. G., R. F. Dell, A. M. Newman. 2004. Optimizing military capital planning. *Interfaces* **34**(6) 415–425, exhibits more than a trillion dollars worth of applications following (C1), (C2), and many of the formulettes.

Brown, G. G., R. F. Dell, and R. K. Wood. 1997. Optimization and persistence. *Interfaces* **27**(5) 15–37, develops (S1) and many more features to preserve similarity, or force diversity between revisions to baseline plans.

Geoffrion, A. M. 1987. An introduction to structured modeling. *Management Sci.* **33**(5) 547–588.

Hamming, R. W. ca. 1985. Author Brown's recollection of a private communication over lunch, but likely something that also appears in at least one of Dick's many legacy publications. Those who knew Dick understand the mandatory exclamation point. We miss him.

Stevens, S. P., S. W. Palocsay. 2004. A translation approach to teaching linear programming formulation. *INFORMS Trans. Ed.* **4**(3) 1–27, http://ite.pubs.informs.org/Vol4No3/StevensPalocsay/.