Optimization and Persistence

GERALD G. BROWN
Operations Research Department
Naval Postgraduate School
Monterey, California 93943-5219

ROBERT F. DELL
Operations Research Department
Naval Postgraduate School

R. KEVIN WOOD
Operations Research Department
Naval Postgraduate School

Most optimization-based decision support systems are used repeatedly with only modest changes to input data from scenario to scenario. Unfortunately, optimization (mathematical programming) has a well-deserved reputation for amplifying small input changes into drastically different solutions. A previously optimal solution, or a slight variation of one, may still be nearly optimal in a new scenario and managerially preferable to a dramatically different solution that is mathematically optimal. Mathematical programming models can be stated and solved so that they exhibit varying degrees of persistence with respect to previous values of variables, constraints, or even exogenous considerations. We use case studies to highlight how modeling with persistence has improved managerial acceptance and describe how to incorporate persistence as an intrinsic feature of any optimization model.

The reasonable man
adapts himself to the world;

the unreasonable one persists in trying to adapt
the world to himself;

Therefore, all progress depends on the unreasonable man.

—*Man and Superman*, George Bernard Shaw

Optimization-based decision support systems, that is, decision support systems built around one or more mathematical programming models, are predominantly employed as follows: A model is used to produce a plan, the plan is pub-

Copyright © 1997, Institute for Operations Research and the Management Sciences
0092-2102/97/2705/0015$5.00
This paper was refereed.

PROGRAMMING—LINEAR—APPLICATIONS
PROGRAMMING—INTEGER—APPLICATIONS

lished, revised data become available and are incorporated into the model, the revised model with many or all of the original decision variables and perhaps some new variables is solved one or more times, and a revised plan is published. This cycle repeats in periodic or continuous review. Confronted with revisions, managers frequently object, "We have committed ourselves to decisions based on prior model advice; don't ask us to change our plans unless we have some compelling reason to do so." New plans that retain the features of prior published plans are more acceptable to decision makers than plans that require drastic changes. We have developed methods for incorporating this kind of *persistence* in modeling linear, mixed-integer, and integer linear programs.

We also use the techniques of persistence to incorporate managerial requests and preferences that arise outside of the cyclic-review process. Sometimes, a manager has useful information about an optimization scenario that cannot be easily incorporated into a model, yet this information is critical for obtaining a usable solution. For instance, forecasted severe weather may affect the production of certain products at a plant next week, but the plant's production-planning model does not encompass weather. To handle this problem, a manager could establish a set of weather-feasible production targets for the affected products, make a model run that is persistent with respect to those targets, and thereby obtain a usable solution.

In our experience, lack of persistence is one of the most common sources of complaints about optimization. Some evidence of our struggles with persistence can be found in our publications on production planning [Avery, Brown, Rosenkranz, and Wood 1992; Brown, Geoffrion, and Bradley 1981; and Brown, Graves, and Honczarenko 1987], dispatching [Bausch, Brown, and Ronen 1995; and Brown, Ellis, Graves, and Ronen 1987], ship scheduling [Brown, Dell, and Farmer 1996; Brown, Goodman, and Wood 1990; and Brown, Graves, and Ronen 1987], capital budgeting [Brown, Clemence, Teufert, and Wood 1991], and global supply chain management [Arntzen, Brown, Harrison, and Trafton 1995]. Over time, we have realized that we need to state models like these differently to formally incorporate persistence. (We use *persistent* to mean "pertaining to persistence.") In addition, in the earliest design of a model we need to consider the cyclic-review process in which the model may be used, and we need to design the models to be flexible enough to handle unforeseeable managerial requests or preferences. We must take into account how the model will be used in the real world.

The literature, for the most part, gives short shrift to the likely real-world use of an optimization model. A refreshing exception is Schrage [1991, p. 129]:

> Multiperiod models are usually used in a rolling or sliding format. In this format the model is solved at the beginning of each period. The recommendations of the solution for the first period are implemented. As one period elapses and better data and forecasts become available the model is slid forward one period.

Unfortunately, Schrage does not go on to point out that a multiperiod model's advice might need to reflect the model's own prior prescriptions. He leaves us with no guidance about how to model and imple-

ment persistence.

The lack of advice on persistence in the literature probably derives from the following:

—Many papers discuss only hypothetical, pilot, or new applications, but problems with persistence usually emerge after a model is used for awhile. (In many of our applications, we have not initially considered persistence, and only over time does this oversight become a nuisance.)

—Modelers write most papers, and they usually focus on how to obtain an optimal solution efficiently, rather than how that solution is going to be used. This focus can bias the modeler to accept the disruptive consequences of an optimal solution because it is, after all, an optimal solution. If managers wrote more papers, the focus might change, since they normally prefer usable solutions over mathematically optimal ones.

—Everybody does it but nobody admits it. Sooner or later, most modelers deal with problems of persistence, and typically they resolve the problems by simply fixing certain variables to their desired values. Few modelers are proud of doing this.

We address these issues by

—Using a series of case studies that demonstrate how persistence can mediate the differences in focus between managers and modelers; and

—Showing how to develop models from the start with persistence in mind.

Scheduling Coast Guard Cutters

The First United States Coast Guard District has used CutS (Cutter Scheduler) to schedule cutters for three years [Brown, Dell, and Farmer 1996]. The mixed-integer linear program within CutS assigns 16 cutters to weekly patrols, maintenance, and training assignments over a calendar quarter while minimizing total time in transit. If changes occur after the Coast Guard has promulgated a CutS schedule to the fleet, it is critical that CutS incorporates persistence in remodeling schedule revisions.

For instance, the summer 1994 schedule—developed with CutS and approved after slight modification by the scheduler and cutter captains—planned on the cutter *Sanibel* being unavailable for three weeks beginning in late July. This unavailability turned out to be delayed by three weeks. Presented with a modification of just one cutter's availability, the nonpersistent version of CutS suggested 52 major changes, where "major change" is defined as the addition or deletion of a week's patrol assignment in some cutter's schedule. These changes influenced 12 of the 13 weeks of the quarter and affected the schedules of 11 of 16 cutters. This solution was mathematically optimal and technically implementable but managerially impractical.

The need for a persistent solution in this case is clear: We want to retain as much of the already-published schedule as practical, and we don't want the cost of the schedule to change significantly. In the persistent version of CutS, we "encourage" binary assignment variables to take on the values they had in the solution corresponding to the published schedule. We do this by converting the original variables into *elastic persistent variables*. Each such variable has a target value it is encouraged to obtain and a linear penalty for any deviation from the target. The conversion is particularly simple for binary variables because it is necessary to modify

only the objective coefficients of the original variables. To keep the cost of the revised schedule close to the original cost, we converted the original objective function into an *aspiration constraint* (a constraint on an "aspiration level," such as mentioned by Mack [1971, pp. 197–200]). Thus, the objective function of the persistent model is a surrogate objective that just measures deviations from the original schedule.

The original schedule for summer 1994 cost 570 transit hours, and we were able to constrain the revision to cost no more (a user-moderated inflation of the original schedule cost can also be used). The surrogate cost of any assignment for the cutter whose availability was changed (the *Sanibel*) is zero. For other cutters, the surrogate cost of assignments in the revision that are identical to those in the original schedule is zero, exchanging patrols with nonpatrols costs 10, and exchanging maintenance and training assignments costs one. For the summer 1994 revision, "Persistent CutS" prescribed only 11 major changes, five of which were for the *Sanibel* (Figure 1).

Revisions such as this are not only more managerially acceptable, they are typically much easier to solve than the corresponding original schedules. Here, the revision is about three times faster to solve than the original.

Base Realignment and Closure Action Scheduler

BRACAS [Dell 1997] is a mixed-integer linear program developed for the US Army to guide it in closing and realigning military installations. Realigning an installation (a military base, for example) means assigning different units to it. BRACAS

	Weeks													Weeks												
Cutter	40	41	42	43	44	45	46	47	48	49	50	51	52	40	41	42	43	44	45	46	47	48	49	50	51	52
	Revision with CutS													Revision with persistent CutS												
Adak					Δ	Δ	Δ	Δ	Δ	Δ																
Wrangell		Δ			Δ		Δ		Δ	Δ																
Sanibel	Δ		Δ					Δ		Δ	Δ		Δ			δ					δ		δ	δ	δ	
Monomoy																δ										
Jefferson	Δ	Δ	Δ								Δ	Δ	Δ								δ					
Grand															δ											
Bainbridge					Δ		Δ		Δ			Δ														
Pt-Bonita					Δ		Δ		Δ	Δ	Δ	Δ														
Pt-Francis	Δ								Δ	Δ		Δ	Δ													
Pt-Jackson			Δ								Δ	Δ	Δ													
Pt-Hannon	Δ	Δ						Δ		Δ	Δ	Δ									δ					
Pt-Turner	Δ		Δ					Δ			Δ													δ		
Pt-Wells																									δ	
Penobscot																										
Sturgeon																										
Thunder																										

Figure 1: The First Coast Guard District revised its approved, published, summer 1994 schedule to accommodate a three-week delay in a three-week unavailability of the cutter *Sanibel*. In an optimal solution, CutS responded with the major changes (patrol reassignments) shown at the left on a Gantt chart with rows representing cutters, columns representing schedule weeks, and the Δ indicating addition or deletion of a patrol week (a major change) in the revised schedule. Persistent CutS reduced major changes to those shown at the right with δ's.

maximizes the expected net present value of savings the army accrues by scheduling expenditures for closure and realignment in each of six planning years. It does this while satisfying a number of constraints governing the way and the rate at which the army must spend money.

In 1995, after a long planning process and many reviews, the Congress approved the army's plan to close 28 installations and realign 13 others to save, eventually, $450 million per year. This approval included a base realignment and closure (BRAC) budget totaling about a billion dollars over six years. The budget was based on cost estimates the army had made without extensive field studies and without using BRACAS.

Next, the army obtained better cost estimates and used them to propose an installation-by-installation budget for each of the six planning years. These proposed budgets covered BRAC expenditures and another billion dollars in separately authorized environmental cleanup costs. Using the original (nonpersistent) BRACAS, the army discovered that, among other things, reallocating $100 million to an earlier phase of the BRAC process would increase savings by $233 million. In late 1995, Congress approved this acceleration, and the army published the year-by-year, category-by-category BRAC and environmental-cleanup budgets.

In February 1996, the army received, from each target installation, revised estimates of annual BRAC and environmental costs. Compilation of the total annual costs derived from these estimates showed budget overruns in early years (Figure 2). Clearly, the yearly budgets had to be revised to be consistent with the amounts approved by Congress.

The yearly BRAC budgets break into four primary categories: construction, environmental cleanup, operating and maintenance, and "other." The army could reallocate BRAC funds among categories within years but not between years. Unfortunately, in these BRAC cost estimates, the target installations provided little guidance about how they might reallocate funds. BRACAS, with enhancements to encourage persistence, provided a model to reallocate yearly BRAC budgets across categories and installations. Priorities based on estimated savings guided the reallocation, while constraints ensured that spending stayed within yearly budget totals.

Within the persistent BRACAS, a *ranged persistent constraint* provides upper and lower limits (ranges) for each target budget category by installation. The installations' planned costs for environmental cleanup covered initial studies and essential preliminary tasks that could not be delayed. We fixed expenditures for these categories, that is, we set upper and lower ranges in the associated persistent budget constraints equal to established values. Construction plans are difficult to change, so if an installation had requested more than a million dollars, BRACAS ranged the reallocation within 10 percent of plan. Operating and maintenance and "other" BRAC costs are somewhat more flexible. We allowed yearly operating and maintenance requests above $2 million to range from an 80 percent decrease to a 150 percent increase. We allowed requests below $2 million to be in-

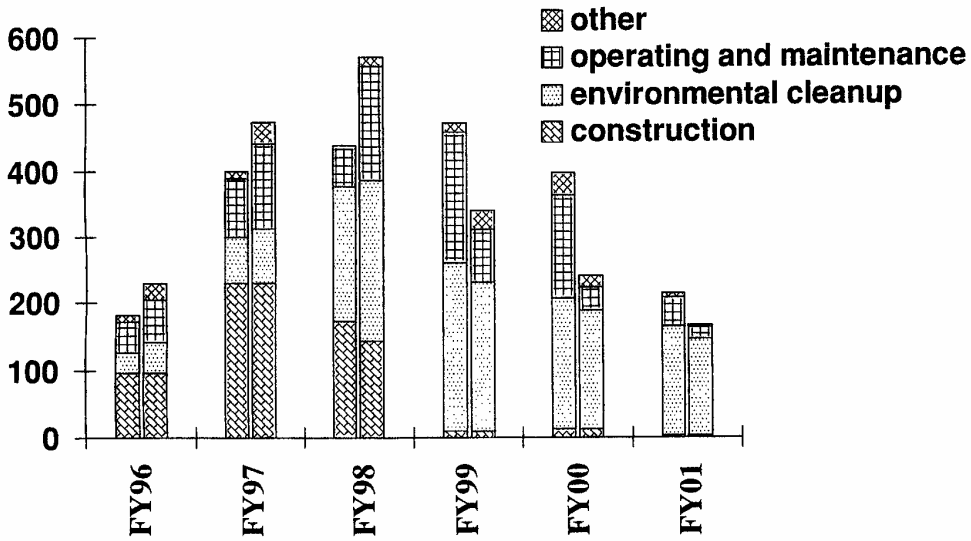


Figure 2: In late 1995, the army published an approved six-year plan for spending about $2 billion to close and realign military installations (left-hand bars). Soon after, the target installations submitted detailed individual schedule revisions that agree with the published plan in total amount, but not in timing (right-hand bars). The army used persistent BRACAS to reschedule the target installations revisions to comply with the plans Congress had approved.

creased to 35 percent of the total six-year operating and maintenance amount requested by the installation. We set the range for "other" requests above $2 million between −90 and +150 percent and permitted requests for lower amounts to increase to 35 percent of total. Persistent BRACAS also constraints budget totals in each year to Congressionally approved levels, exactly.

The army is following the persistent BRACAS advice.

Strategic Analysis of Integrated Logistics Systems

SAILS is an integrated decision support system for building, modifying, solving, maintaining, and interpreting large-scale strategic multicommodity logistics network design models [INSIGHT 1994]. SAILS has been used for more than 20 years by scores of companies, including about half of the Fortune 50, their consultants, and a number of university logistics programs. Geoffrion and Powers [1995] review its remarkably long history.

The databases and models that underlie SAILS are typically large (for example, hundreds of millions of freight rates and millions of model variables), but a modest number of entities are important to managers: These govern the go/no-go structural decisions (Figure 3). SAILS has been so successful largely because of its persistent features. Managers can use these features to ensure that a new network design does not depart very much from an old design and that it is based on their guidance.

SAILS' graphical user interface cloaks huge amounts of detail and offers the user

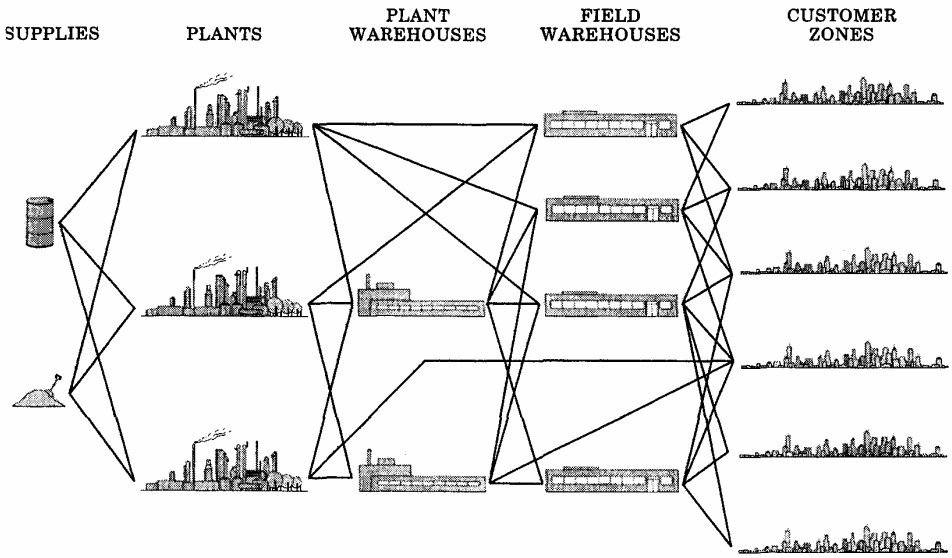


Figure 3: A typical SAILS logistics network is best illustrated in terms of physical system entities (figure adapted from Geoffrion and Powers [1995]). An optimal design can be principally expressed by distinguishing which of these entities are to be open and which closed. Hidden are millions of underlying details from sole sourcing to product recipes—details that are essential but not likely to be the principal focus of decision makers.

intuitive ways to influence network-design decisions. One can fix model variables to open or close suppliers, plants, equipment, packing lines, conversion recipes, distribution centers, product bundles, sole sourcing, and so forth. Fixing variables is the simplest and strongest way to insure persistence. Customer demands for products can be scaled, eliminated, patterned after guide forecasts, and aggregated via a host of georeferents. One can restrict commodity flows to automatically follow the patterns, but not necessarily the amounts, of some prior solution. *Elastic persistent (ranged) constraints* govern throughput limits: These are constraints that one can violate at some linear penalty cost per unit violation above or below the target ranges. (*Elastic persistent (equality) constraints* are a special case of the ranged variety.)

Managers can guide SAILS by fixing certain decisions and in other ways. For instance, for any set of candidate open/close decisions, one can specify the minimum and maximum number of "opens," that is, limits on the set cardinality of the open/close decisions. Using these two features, managers can put into effect such statements as: "I don't know where we'll relocate all of the distribution centers, but we'll be in Columbus, Atlanta, and Denver, and we'll operate no more than 20 locations."

SAILS invites no-fault recommendation of a solution from past experience. One can dredge up this advice from detailed records of a past design, and perhaps subject it to some qualitative editing via interface push buttons. Or the advice may be

less detailed and expressed in such managerial terms as "the Reno plant will likely still stay open; try it" or "likely not." One can give preference to any decision, but it is not required. The interface sorts out what this advice means to successive or competing models, which do not necessarily share any common constraints or variables but presumably have a "Reno plant."

SAILS always follows this advice first in a *persistent preemptive enumeration,* which is a modification of standard branch-and-bound enumeration that incorporates and exploits persistence. Because of its elastic features, SAILS guarantees completion of at least one global design that follows all the advice provided to the letter. Then, armed with this initial incumbent solution, the advice that suggested it, and limits on what digressions are permitted, SAILS continues to make system improvements in a customary enumeration.

Although the initial guidance alone may not produce an incumbent solution with acceptable quality, the suggestions and the bounds that they contribute can accelerate subsequent enumeration. In fact, good advice can speed up SAILS by an order of magnitude or more. It's also comforting to know that whatever SAILS finally suggests, it has given management guidance primary consideration.

Hamming Distance and Submarines

The sum of the absolute values of the bit-by-bit differences between two binary vectors is called the *Hamming distance* between them [Hamming 1986, p. 45]. Suppose each binary variable in a set represents the decision to set up production on a machine next month or deploy a ship next week or change customer sourcing next year or move a berthed submarine tomorrow [Brown, Cormican, Lawphongpanich, and Widdis 1977] (Figure 4). The Hamming distance between a published binary plan and a successive revision provides a simple but useful gauge of the turbulence or lack of persistence between the two proposals.

We use ranged persistent constraints to limit the Hamming distance between successive solutions and call these constraints *Hamming cuts.* Alternately, we sometimes penalize turbulence by placing the Hamming distance in the objective as a *Hamming penalty.* We implement this penalty by using an elastic persistent constraint whose goal is to avoid any change between solutions. Both Hamming cuts and Hamming penalties use a simple linear functional: The coefficient of each binary variable is 1 if its prior published value is 0, and it is -1 otherwise.

Kellogg Planning System

The Kellogg Planning System (KPS) is an unpublished model that relies on a large linear program to determine the production and distribution of cereals and convenience foods at a weekly level of detail. KPS models processing facilities producing base products, packaging lines converting base products into finished stock keeping units (skus), and shipments among and inventory within processing locations and distribution centers (DCs). The object is to meet demand at minimum cost.

KPS encompasses about a hundred base products, several hundred skus, about a dozen producing locations and about a dozen DCs. This is a big, highly detailed

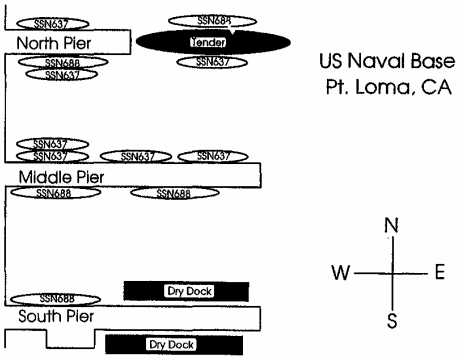


Figure 4: A plan for submarines berthed at US Naval Base, Pt. Loma, California shows seven SSN637 Sturgeon and five SSN688 Los Angeles class submarines situated to receive shore services on a given day. Each submarine needs different shore services day by day, and each berth position, including a tender ship that can act as a berth when moored as shown, has differing abilities to render such services. The navy minimizes berth shifts: Moving multibillion dollar submarines is time consuming and interrupts services. However, new port arrivals, departures, and daily service schedules make some berth shifts unavoidable. An optimization-based planner has needed some guidance to minimize berth shifts that only superficially improve service benefits. The berth-planning model expresses the location of each submarine on any given day as a binary decision variable and limits undesired berth shifts between days by penalizing the Hamming distance between existing or planned positions and suggested shifts of these positions.

model even though it does not explicitly deal with raw materials. Planners solve a 20-week tactical "production model" every Sunday morning, with planning week one beginning the next day. There is also a strategic what-if version of KPS, with monthly detail, that they use to evaluate potential major changes in production capacity, inventory policy, and so forth.

KPS is persistent in several ways.

On the rolling weekly horizon, raw materials and packaging materials required for week one will already be arranged, so KPS cannot change week-one production or packaging decisions. Also, "pending stock orders" restrict week-one shipping decisions. So, persistent variables freeze (fix) most production and distribution decisions in week one.

Lead times for some materials exceed one week, so in some cases the system imposes a partial freeze of the packaging schedule in weeks two and beyond. It also uses a partial freeze of other activities to incorporate management knowledge or even hunches about future conditions that a model cannot guess. For instance, it can reserve scarce production capabilities for key products in critical weeks by withholding them from products with other alternatives.

Strategic KPS employs an 18-month horizon to evaluate such issues as locating new production facilities or realigning existing capacity. Forecasting demand 18 months ahead is not easy, especially when sales promotions create spikes of highly uncertain size. KPS responds optimally, in a mathematical sense, to estimated demand spikes and will even anticipate those far in the future by adjusting decisions in early months. Because of this, managers initially found KPS too "nervous." In particular, from run to run, KPS prescribed significant changes in production across time, locations, and products when planners changed far-term demand estimates for only a few products.

"Nervousness" is just a lack of persistence caused, in this case, by KPS's "omniscience." This omniscience is mitigated, in

practice, by employing a problem cascade (for example, Brown, Graves, and Ronen [1987] or the "subhorizons" suggested by Charnes and Cooper [1961, p. 370]). In particular, the strategic version of KPS uses a sliding window of five months, first optimizing months one through five, then fixing month-one variables and optimizing months two through six, then fixing month-two variables, and so forth. Unlike most cascade schemes, this one never solves the entire (18-month) problem.

New plans that retain the features of prior published plans are more acceptable. . . .

Making KPS myopic has several advantages. Managers like the results: KPS doesn't anticipate spikes and rearrange production plans too early. Myopia also eliminates the need to explicitly model the shelf life of perishable inventories because KPS doesn't produce to inventory until it sees demand, and five months is a reasonable shelf life. The myopic model is also much smaller and faster to solve.

Helicopter Fleet Planning

The PHOENIX optimization model helped the US Army to prevail over budget critics and modernize its aging post-Vietnam helicopter fleet [Brown, Clemence, Teufert, and Wood 1991]. The army has used PHOENIX and its progeny to plan modernization of helicopters as well as a variety of other equipment fleets. The successes of PHOENIX, reinforced by intense scrutiny during defense budget debates, have also given other authors the confidence to apply optimization to other military-procurement and equipment-management problems. Illustrative examples are presented by Dundas [1996], Faircloth [1989], and Staniec [1996]. PHOENIX-like models are now influencing planned expenditures of many billions of dollars, and persistence is an important feature of most.

PHOENIX-like models are all multiperiod models that address long-term equipment replacement issues, including optimal timing of major maintenance, refit, retirement, and most important, new procurement. Military procurement is distinguished by high fixed setup costs: Research, development, testing, and evaluation costs are large, and production costs are amplified by first-time application of high technology, security, and limited production quantities. Alternate candidate production and renovation runs are arranged as campaigns, each with a start date, duration, and level of effort. This partitions production methods to isolate cost-estimation, economies-of-scale, and learning-curve effects. Military equipment also must be fielded in compatible units of inter-operating equipment types. This augurs well for optimization.

PHOENIX uses many elastic persistent constraints and variables to encourage desired spending levels, average fleet age, average "technological advantage" of the fleet, and so forth. (The persistent features we discuss have all been used in PHOENIX, but not all of these appear in the simplified published model.) *Discount rates* reduce persistent penalties in each period, typically a year, to net present value or even lower, reflecting the army's uncertainty about the future and a reasonable desire to delay violations as long as possi-

ble. PHOENIX's time horizon covers many years, so this is important. Of course, net present value is not a precise concept when dealing with nonmonetary units, such as technological advantage, but all elastic penalties are usually adjusted by the same discount rate, the rate used for money.

PHOENIX keeps track of individual helicopter age and not only limits it, but also uses an elastic persistent constraint by year on the average age of the entire helicopter fleet: "Try to keep the average operational helicopter no older than T years." This is an example of a weighted average elastic persistent constraint the system uses to smooth or moderate fluctuations in plans to conform with some overall expectation.

Some of PHOENIX's persistent constructs are elastic cumulant persistent constraints with cumulant target values or ranges. These constraints represent our desire for a sum of events to meet a sum of subtargets since some base event. For instance, consider a persistent model using yearly retirement variables and targets for retirements. Now, the sum of yearly targets from the beginning of the planning horizon to the end of each year t does imply a set of cumulant targets. However, such a model would allow more-or-less independent deviations from its yearly targets. Thus, it might miss the implied cumulant targets by significant amounts, especially toward the end of its time horizon. In contrast, PHOENIX with explicit cumulant constraints compares cumulative retirements in each year to targets representing total desired retirements from the start of the planning horizon. So, PHOENIX pays penalties in each year for the total deviation since the beginning of the planning horizon and is motivated to keep cumulative retirements on track in every year. (Leachman, Benson, Liu, and Raar [1996] describe another example of cumulant targets.)

An alternative to writing cumulant constraints with yearly variables is to rewrite the model in terms of *elastic cumulant persistent variables*. For example, the cumulant formulation described above could be rewritten to use elastic persistent variables representing cumulative retirements, each with a cumulant target. Any constraint needing retirements in a single year could be written in terms of the difference between cumulant variables in that and its predecessor year.

Persistent Partitioning

A variety of important business, engineering, and scientific applications employ set-partitioning models. Anbil, Gelman, Patty, and Tanga [1991]; Eben-Chaime, Tovey, and Ammons [1996]; and Thuve [1981] provide illustrative examples. These models have a deceptively simple appearance but offer powerful modeling capabilities. Although set-partitioning models can be solved with customary linear integer-programming methods, they are not easy to solve reliably, and they have only become more fashionable as the trustworthiness of solution methods has improved. Predictably, real-world experience reveals issues of persistence, in particular, with respect to incorporating guidance from the user.

As an illustration, suppose there are several hundred packages on a loading dock, each ready to be shipped to its own

destination. A fleet of identical trucks is available, each of which can deliver a truckload of packages to their destinations in some order using a route that cannot exceed some maximum driving distance or time. The problem is: How should the packages be consolidated into a minimum number of feasible truckloads?

In a set-partitioning model for this, we define a constraint for each package to make sure it gets delivered exactly once. We define a binary variable for each candidate truckload, with a unit coefficient in each constraint for a package in that truckload. All we need do is select the minimum number of binary variables, that is, columns, so that each constraint has exactly one unit coefficient selected (Figure 5).

binary variable															
a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	...	
		1			1	1									= 1
	1			1						1		1			= 1
1	1							1	1			1			= 1
1	1						1		1	1	1				= 1
			1	1		1							1		= 1
1					1	1	1								= 1
		1			1										= 1
		1	1									1	1		= 1
	1		1					1	1		1		1		= 1

Figure 5: A set-partition puzzle: Select a set of columns in this matrix so that there is exactly one selected unit element in each row. (Minimizing the number of such columns is no easier.) An illustrative application views each row as the requirement to deliver a package and each column as an alternate delivery route. (We give a hint in the text and later a solution.)

Of course, the real world is more complicated. For example, Bausch, Brown, and Ronen [1995] describe a freight consolidation case that has a number of necessary embellishments. Trucks are not identical, and the cost of a delivery route is a complicated function of which packages a truck carries and when and where it must deliver them. Most of these details are extrinsic. That is, they govern the generation of cost coefficients and the locations of unit column coefficients for variables but otherwise do not appear explicitly in the model. This is a blessing and a curse.

The number of binary variables (in this example, the number of subsets of packages forming candidate truckloads) can be enormous. Often a key to success is a sampling mechanism that can generate from this huge population a restricted subset of columns from which a good partition can be found (for example, a good set of routes delivering all the packages). Barring exceptionally good fortune, restricted partition solutions will exhibit some flaws, such as ridiculously high cost or outright infeasibility (for example, packages not delivered). At this point, the restricted partition needs help.

One approach to improving the solution is to return to the column generator with "internal," model-provided advice about what kind of columns (routes) are needed and to generate such columns. (Graves, McBride, Gershkoff, Anderson, and Mahidhara [1993] use this device in scheduling airline crews.)

Another approach accepts external guidance from an experienced user, who can review a tentative solution and decide when it's reasonable to change the problem by simply delaying a delivery, making a special delivery, hiring an outside delivery service, and so forth. A phone call relaxing a bottleneck beats a clever algorithm every time.

But, if an experienced user contributes time and advice to deal with a flaw and then reoptimizes, it's not a good idea to capriciously create new flaws. So, reoptimization with human assistance makes persistent techniques essential.

When reoptimizing, the user should be able to suggest that the previous partial solution be incorporated where it appears to be sound. One way to do this is by simply fixing variables. Fixing variables isolates the parts of the solution that look good, and fixing a variable to *1* can break up a set-partitioning problem into smaller, more easily solved pieces. (For instance, you will find it easier to solve the puzzle in Figure 5 if you follow the hint that column *h* is part of a partition.) However, we prefer to employ elastic persistent variables or constraints for this problem so that if the model does not select user-preferred routes, a penalty is inflicted. (Brown, Goodman, and Wood [1990] add elastic persistent constraints to their generalized set-partitioning model for annual scheduling of the US Atlantic Fleet.) Since the set-partitioning problem is a binary integer program, penalizing nonuse of preferred columns is much like using a Hamming penalty as part of the objective.

Persistence Is Not New

Persistence is not new, but the literature is scant. Researchers addressed the basic issue of persistence as early as the 1950s and have published recent research on issues of implementation.

Charnes, Cooper, and Ferguson [1955] describe a linear-programming model to determine a consistent linear formula for executive compensation. (Charnes and Cooper [1961, Chapter 10] later restate this case.) The model selects weights for employee-compensation factors so that the resulting formula for executive salaries meets company-specified criteria "as closely as possible." Examples of these criteria are (1) higher-ranked employees should be paid more than lower-ranked employees and (2) salaries should be competitive on an industry-wide basis. The model employs both ranged and elastic persistent constraints.

Charnes and Cooper [1961, p. 215] start and end their discussion of goal programming with advice about persistence. They seek solutions such that "long-run considerations are not obliterated by immediately attainable objectives" and conclude:

> For example, constraints might be entered to demarcate regions which are "good enough," and the objective restated to ensure that programs are either (a) good enough or (b) close to good enough, etc.

Ranged persistent constraints can be used to implement "good enough" and ranged elastic persistent constraints can be used to implement "close to good enough."

Bowman [1963] demonstrates that one can incorporate management's past decisions to produce effective present decisions using a linear decision rule. A referee's comment quoted in that paper states "That managerial decisions might be improved more by making them more consistent from one time to another than by approaches purporting to give 'optimal' solutions to explicit cost models . . . especially for problems where intangibles (run-out costs, delay penalties) must otherwise be estimated or assumed." Our enthusiasm for this view is guarded. An unfortunate thrust of this comment, Bowman's

work, and extensions of his work [Hurst and McNamara 1967; Jones 1967; Kunreuther 1969; and Bowman 1984] is to emphasize subjective considerations and de-emphasize or ignore objective optimization altogether. Persistence can synergistically combine subjectivity and objectivity.

A number of authors have suggested constraints to incorporate subjective considerations into mathematical models. Huysmans [1970] suggests adding "human constraints" to operations research models. Trull [1966] recounts the advice of organization theorists March and Simon [1958]: Once an initial decision is reached, it establishes decision rules or procedures that constrain future decisions. Little [1970, p. B-483] states that subjective judgments must be incorporated into models used by managers because people have a way of making better decisions than their data seem to warrant. He concludes:

> The model is meant to be a vehicle through which a manager can express his views about the operations under his control. Although the results of using the model may sometimes be personal to the manager because of judgmental inputs, the researcher still has the responsibilities of a scientist in that he should offer the manager the best information he can for making the model conform to reality in structure, parameterization, and behavior.

All of the subjective considerations we discuss above can be implemented using persistent constraints.

Urban [1974] surveys over 150 articles from the "Application" section of *Management Science* and concludes that management scientists are not building good models from the decision maker's point of view. He reminds us that a manager's most cherished prerogative is to make decisions, and we must take special care to show that the model will supplement and not replace the manager in his or her decision making. For a model in continuous use, Urban also discusses the need to refit any new data and update any assumptions. These requirements are the goal and guide of persistence.

Lewandowski and Wierzbicki [1989] edited a series of papers that describe decision support systems based on "reference-point optimization" and applications of these systems in Poland. Reference-point optimization incorporates managerial requests and preferences within the decision support systems—one of the goals of persistence. The systems interactively form multiple objectives based on user-supplied reference points or aspiration constraints. Their description of aspiration constraints is so generic that we interpret them to include both persistent variables and constraints. However, none of the edited papers highlight the benefit or necessity of persistence, and none illustrate how persistence might be useful in successive model revisions.

Mulvey [1993] diagnoses trouble with optimization models that rely on noisy input data and prescribes a technique he calls robust optimization. Robust optimization (see also Mulvey, Vanderbei, and Zenios [1995]) seeks a solution that, over many potential alternate input scenarios, is close to optimal (solution robust) and almost feasible (model robust). From our perspective of persistence, robust optimization seeks a baseline solution that will persist as best possible with a number of alternate forecast revisions. This is a laud-

able goal but fraught with challenges. We agree with Mulvey: How do you forecast future revisions before you solve the model? By contrast, persistent modeling uses experience as a guide and invites guidance for dealing with change, rather than depending upon precise predictions.

We must take into account how the model will be used.

General nonlinear-programming methods and (linear and nonlinear) decomposition methods solve models indirectly by iteratively making local estimates of directions of improvement and taking steps in those estimated directions. It is wise to be cautious about step length because the neighborhood over which you can trust each local approximation is usually limited by ignorance of the global properties of the problem, and the consequences of prediction error can be serious.

Accordingly, nonlinear-programming algorithms are customarily governed by *trust regions* for variable values over which the local approximations are assumed to have sufficient validity (for example, see Fletcher [1981, p. 207]). Some decomposition algorithms also use trust regions. There are both theoretical and heuristic arguments that such moderation is a virtue, and real-world computational experience is convincing [Brown, Graves, and Honczarenko 1987; Moré 1983].

Even simple linear programs need trust regions because absolute linearity just doesn't hold in the real world. Unless we apply common sense (and perhaps Taylor's approximation) and limit the region over which we can expect linearity (and perhaps Taylor's second- and higher-order remainder terms to remain insignificant), we invite trouble. Ranged persistent variables are variables bounded within desired ranges; the corresponding bounds constitute a hyper-rectangular trust region for primal variables. We have not presented a case study with this ubiquitous persistent feature, but we always use it.

We have also suggested the use of persistent elastic constraints: The associated elastic penalties constitute a *trust region for dual variables.* The proximal terms used in some nonlinear programming algorithms [Kiewel 1985; Mifflin 1977] and in some decomposition algorithms [Ruszczyński 1986, 1993] are similar persistent controls.

We can interpret linear regression models in the light of persistence: Given a set of observations or targets, a corresponding set of independent variables, and a specified metric, find a parametric function of the independent variables that tracks as closely as possible with the targets. In other words, find a parametric function of the independent variables that is persistent with respect to the desired values, the targets. Given this interpretation, we can compare the advantages and disadvantages of two standard regression models.

The least-squares (L2) model is the regression model that people habitually adopt as much for its ease of computation as for the wealth of elegant statistical results deriving from the usual assumptions of homoskedastic normality of observation errors [Draper and Smith 1966]. By contrast, an absolute-value (L1) regression requires one to solve a linear program [Barrodale and Roberts 1973], deal with nonunique solutions, and interpret the sig-

nificance of results without as much statistical support. An L1 regression is not without advantages, however.

L1 regression models are essentially elastic persistent-constraint linear programs. These LPs determine model parameter values that minimize the average absolute deviation between observed response values and forecasts of these. There are textbook examples [Schrage 1991, p. 255]. In an L1 regression, it is easy to add side constraints on estimated model parameters, recourse for missing data, (elastic) limits on maximum estimation error for any observation, and any number of linear embellishments. The resulting model is still a linear program, and our experience shows that a constrained L1 regression is seldom much harder to solve than its unconstrained counterpart. On the other hand, a standard L2 regression problem is an easy-to-solve unconstrained quadratic program, but adding linear constraints changes it into a constrained quadratic program that is more difficult to solve.

Persistence Has Its Costs

Persistent modeling requires little additional data, but you must prepare this data carefully. An elastic persistent variable needs a target value and nonnegative penalties for deviations above or below this target. This can be generalized to allow different per-unit penalties in different ranges, but in practice a single target and simple linear deviation penalties usually suffice and do not require additional constraints. The persistent-variable penalties can constitute a distinct objective or be weighed with other objectives.

A ranged persistent constraint requires a target range governing its value. Ranged elastic persistent constraints also need per-unit linear penalties for violations above and below the target range. Such constraints can also be generalized to allow different per-unit penalties in different ranges, but we find that linear penalties with a single range usually suffice.

The challenge is to express persistent features, especially penalties, to create an insightful, unified model.

Conclusions

Optimization models respond unpredictably to seemingly inconsequential changes in input. This is especially troublesome when an optimized plan has been adopted and inevitable small changes in data necessitate a revision. The model user desires a revised plan that is "not too different" from the old, but reoptimization may prescribe massive revisions completely out of proportion to the changes in inputs. At best, this is annoying. At worst, good models lose credibility with their users.

Our prescription for this problem is *persistence* from the root *persist:*

> 1. to continue steadily or firmly in some state, purpose, course of action, or the like, esp. in spite of opposition, remonstrance, etc.: to persist in a belief; to persist in one's folly. [Webster's 1989]

By making a model *persistent*, a new solution may be obtained that is not too different from the previous solution yet is nearly optimal with respect to standard criteria. The persistent techniques that we use are summarized below.

—A subset of a model's variables may be fixed or frozen to their preferred values. (*Preferred* means "previous" or "desired"

depending on how persistence is interpreted in a given model.) A hyperrectangular trust region for the model variables, centered around a preferred solution, implements another simple form of persistence.

—Some variables can be converted into elastic persistent variables that incur linear penalties for deviating from their preferred values. When binary variables all incur the same penalty for changing from their preferred values, the total penalty measures the Hamming distance between the preferred and new solutions.

—Ranged persistent constraints ensure that certain aggregate quantities do not deviate too far from preferred values. These quantities may or may not be part of an originating nonpersistent model. Ranged persistent constraints with binary variables can limit the Hamming distance of a new solution from a preferred solution; we call such constraints "Hamming cuts."

—Ranged persistent constraints can be too strict. To allow a persistent model more flexibility, we employ elastic persistent constraints that may or may not be ranged. Such constraints encourage aggregate quantities to achieve preferred values or ranges of values, but allow penalized deviations to occur, too. A discount factor is often applied to penalties on such constraints (or variables) when they are indexed by time or proximity.

—Sometimes we convert an original objective function into an aspiration constraint in a persistent model. This is just a persistent constraint, elastic or ranged, that encourages or forces a new solution not to deviate too far in cost from a preferred solution's cost.

—For the purpose of introducing subjective judgments into a model, we sometimes limit the number of binary variables that can be set to one or zero using set cardinality constraints.

—Rather than penalizing or restricting deviations of variables or constraints from previous values, it may be desirable to penalize or restrict deviations that accumulate over time. For instance, we may not want to retire exactly 10 aircraft in each year t in a model, but we might like to see that about $10t$ aircraft are retired on average year-by-year through year t in the model. In such a case, the persistent constructs described above may be applied to cumulant variables or cumulant constraints. When elastic penalties are used with a sequence of cumulant variables or constraints, they imply penalties on a weighted average of their noncumulant counterparts. Weighted average constraints can also directly govern linear combinations of arbitrary performance measures.

There are also two rather different techniques that we use to achieve persistent behavior in certain models:

—By solving a time-indexed model with a problem cascade, that is, by sequentially solving such a model over a subset of the model's time periods (say, 1 through t, 2 through $t + 1, \ldots, T - t + 1$ through T), we induce a t-period myopia in the model that reduces "nervousness" of solutions to minor changes in data.

—A persistent preemptive enumeration can be used to solve mixed-integer programs by branch and bound: The solver investigates preferred solutions before

widening its focus and considering less (subjectively) desirable solutions.

Textbooks, even those with case studies, don't offer much advice about persistence in optimization models. Thus, we have presented a collection of case studies that motivates the need for persistence in such models and shows how to incorporate it. These case studies reflect a long, slow awakening on our part that persistence should be a model enhancement, rather than a crippling oversight. A persistent solution is a more usable solution and, despite the need to solve a larger model, often takes less time to compute than its nonpersistent counterpart.

Persistence is usually added to an optimization model too late, after it misbehaves, and almost all optimization models eventually do misbehave. After the fact, repairing the model and regaining the faith of sponsors can be complicated. It is better to plan for persistence from the outset. To this end, we have provided new vocabulary and new mathematical notation that should help simplify describing and exploring the issues of persistence in optimization models. This vocabulary is defined as used in the body of the paper and is summarized above; the mathematical notation is defined as part of a helicopter fleet-modernization model described in the appendix.

(In Figure 5, columns c, e, h, and i form a partition.)

Acknowledgments

Some of this work has been supported by the Air Force Office of Scientific Research and the Office of Naval Research. We thank Professor Rick Rosenthal (Naval Postgraduate School) for his careful review. We keep examples of Mary Haight's editorial markups handy to show our thesis students that we, too, are humbled by such.

APPENDIX: A PERSISTENT FORMULATION, PHOENIX REDUX

A simplified PHOENIX-like prototype (Figure 6) demonstrates persistence in an initial model formulation. This example does not incorporate all the persistent features we have identified, but it is a good example of how persistence can be expressed concisely to yield an easily understood formulation. Our persistent formulation extends the NPS (Naval Postgraduate School) standard format, a format we enforce on ourselves, our clients, and our students. (Similar formulation formats appear in the literature.) NPS format follows a define-before-use subdivision of model entities, including indices (dimensions), data (and units), variables (and units), and the model's objective and constraints. Typically, we follow this subdivision with verbal descriptions of the objective function and constraints.

We denote elastic persistent variables by a "˘" but otherwise do not distinguish them to simplify the model presentation—the persistent data requirements are deferred to a later section of the formulation as shown in Figure 7. For instance, each continuous inventory variable is introduced as $\breve{X}_{tac}$, for $c < t$, and amplified later to include an associated target value X^0_{tac} and penalties per aircraft under (PX^-_{tac}), or over this target (PX^+_{tac}). For this model, only the production and inventory variables are persistent. Their targets might represent values obtained from a previous run using slightly different data.

Persistent variables (see Figure 8 for a pictorial representation) can be directly accommodated by specialized solution methods (Brown and Olson [1996] and Fourer [1985, 1988, 1992]). Lacking these tools, conventional methods can be used with

Indices:

p = production line,

a = aircraft type,

t = planning year,

c = cohort year (year of manufacture).

Data:

(Note: All costs include an appropriate discount factor based on t.)

B_t = budget available in year t,

FR_t = total aircraft required in year t,

FA_t = desired average age of all aircraft in year t,

RC_{tac} = cumulative number of aircraft of type a, cohort c to be retired by year t,

α_a = annual survival fraction of aircraft a.

l_a = lag in years between the year when air-craft a is paid for and the year it joins the fleet,

PC_p = cumulative aircraft to be produced in a production campaign on line p,

CP_{at} = the unit cost of producing aircraft a in year t,

CR_{tac} = the cost of retiring aircraft a, cohort c, in year t,

CO_{tac} = annual operating and maintenance cost for an aircraft a of age $t - c - l_a$ years,

$CF_{ptt'}$ = the fixed cost paid in year t when line p is started in year t',

λ = relative weight for standard costs versus linear persistent elastic penalties

Variables:

$\check{X}_{tac}$ = for $c < t$, inventory of operational aircraft of type a of cohort year c in year t ($\check{X}_{tat}$ is the number of aircraft a produced in year t),

R_{tac} = number of aircraft a in cohort c that are retired at the beginning of year t,

Y_{pt} = 1 if production line p is opened at the beginning of year t, 0 otherwise.

Model ($\sqcup$ and $\vee$ indicate persistent features): minimize

$$\lambda\left(\sum_{a,t>c} CO_{tac}\check{X}_{tac} + \sum_{tac} CR_{tac}R_{tac}\right)$$

$+ (1 - \lambda)$ (linear persistent elastic penalties)

subject to

$$\sum_{ac} \check{X}_{tac} \overset{\sqcup}{=} FR_t \qquad \forall t \quad (1)$$

$$\sum_{ac} (t - c - l_a)\check{X}_{tac} \overset{\vee}{\leq} \sum_{ac} FA_t\check{X}_{tac} \qquad \forall t \quad (2)$$

$$\sum_{a} CP_{at}\check{X}_{tat} + \sum_{ac} CO_{tac}\check{X}_{tac} + \sum_{p,t'\leq t} CF_{ptt'}Y_{pt'} + \sum_{ac} CR_{tac}R_{tac} \overset{\vee}{\leq} B_t \qquad \forall t \quad (3)$$

$$\sum_{ta} \check{X}_{tat} \overset{\vee}{\leq} \sum_{t} PC_pY_{pt} \qquad \forall p \quad (4)$$

$$-\alpha_a\check{X}_{t-1,ac} + \check{X}_{tac} + R_{tac} = 0 \qquad \forall t, a, c \quad (5)$$

$$\sum_{t'\leq t} R_{t'ac} \overset{\vee}{\leq} RC_{tac} \qquad \forall t, a, c \quad (6)$$

$$\check{X}_{tac} \geq 0,\ R_{tac} \geq 0 \qquad \forall t, a, c$$

$$Y_{pt} \in \{0, 1\} \qquad \forall p, t$$

Additional data for persistence follows in Figure 7.

Explanations for the objective function and constraints follow in Figure 7.

Figure 6: An example of PHOENIX, the model used to modernize the army's helicopter fleet, with modifications for persistence. Conventional notation is used with persistent features distinguished by " $\vee$ " or "$\sqcup$". Elastic persistent variables (with a " $\vee$ " over the variable) have a target value, and linear penalties for deviations from this in either direction. Ranged persistent constraints (symbolically, $\overset{\sqcup}{=}$) may vary in value over a limited range. Elastic persistent (ranged) constraints (symbolically, $\overset{\vee}{\leq}$) signify constraint ranges that can be violated at a linear penalty per unit violation.

the addition of some auxiliary variables. For instance, a continuous persistent variable $\check{X}$ may be expressed as the sum, $X^0 + X^+ - X^-$, with X^0 the fixed target, X^+ a positive deviation costing PX^+X^+, and X^- a negative deviation costing PX^-X^-. The representation is even simpler for binary variables.

Ranged persistent constraints in Figure 6 are denoted $\overset{\sqcup}{=}$, signifying that a range will be provided. For example, each instance of constraint (1) superficially stipulates that FR_t aircraft must be procured; however, the notation "$\overset{\sqcup}{=} FR_t$" alerts us that a persistent constraint range is implied, and in the following persistence section of the formulation, this is amplified to be a number no less than $\underline{FR}_t$ and no greater than $\overline{FR}_t$. Ranged constraints are supported by virtually all commercial

Additional data for persistent model features

Data for elastic persistent variables:

(Note: All penalties include an appropriate discount factor based on t.)

X^0_{tac}, PX^+_{tac}, PX^-_{tac} target and penalties per unit above and below target for $\check{X}_{tac}$, (X^0_{tac} is production target when $c = t$ and is an inventory target when $c < t$.)

Data for ranged persistent constraints:

$\underline{FR}_t$, $\overline{FR}_t$ allowed range for FR_t.

Data for elastic persistent constraints:

$\underline{FA}_t$, $\overline{FA}_t$, PFA^+_t, PFA^-_t range and penalties per unit above and below range for FA_t,

$\underline{B}_t$, $\overline{B}_t$, PB^+_t, PB^-_t range and penalties per unit above and below range for B_t,

$\underline{PC}_p$, $\overline{PC}_p$, PPC^+_p, PPC^-_p range and penalties per unit above and below range for PC_p,

$\underline{RC}_{tac}$, $\overline{RC}_{tac}$, PRC^+_{tac}, PRC^-_{tac} range and penalties per unit above and below range for RC_{tac}.

Explanation of objective function and constraints

Objective: Minimize a weighted sum of operating and retirement costs plus linear persistent elastic penalties.

Eqn. (1): Total aircraft inventory must fall within a given range in each year.

Eqn. (2): Average fleet age should fall within a desired range in each year.

Eqn. (3): Expenditures in each year should fall within acceptable budget ranges.

Eqn. (4): Cumulative production on each opened line should fall within efficient ranges.

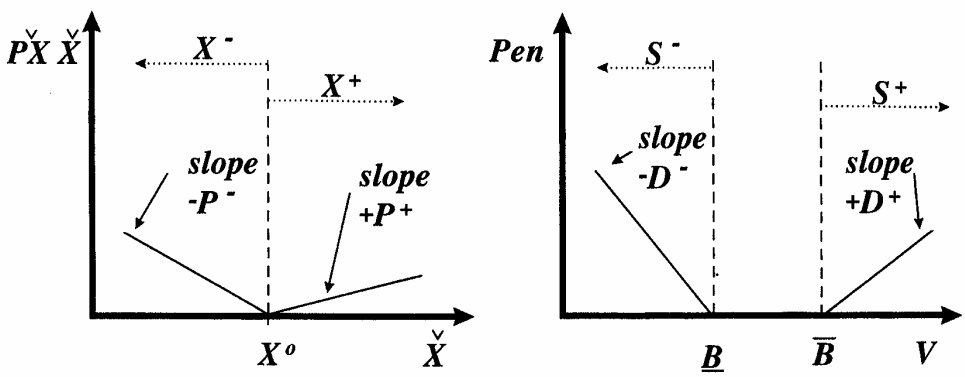
Eqn. (5): Production and attrited inventory must balance between years.

Eqn. (6): Cumulative retirements should fall within desired ranges each year.

Figure 7: Additional data underlying the persistent PHOENIX reformulation, and an explanation of the objective function and constraints.

optimizers and can be also be formulated as a standard equality constraint with a bounded slack variable.

The elastic persistent (ranged) constraints ($\underline{\vee}$) indicate that a range and penalties for violating the range will be provided (Figure 8 pictorially represents this). Specialized linear programming algorithms handle elastic constraints directly [Brown and Olson 1996], but any such



elastic persistent variable: $\check{X}$
implementation: $\check{X} = X^0 - X^- + X^+$

elastic persistent constraint: $V \underline{\vee} B$
implementation: $\underline{B} - S^- \le V \le \overline{B} + S^+$

persistent penalty: $P\check{X}\check{X}$
implementation: $P\check{X}\check{X} = P^- X^- + P^+ X^+$

elastic penalty: Pen
implementation: $Pen = D^- S^- + D^+ S^+$

Figure 8: In this pictorial representation of elastic persistent variables (left), and elastic persistent constraints (right), the variable V could represent a single variable but likely represents a more complex constraint value, such as $\sum_{t' \le t} R_{t'ac}$ in equation (6) of Figure 6.

constraint can be represented for standard solvers as an equality constraint with penalized artificial variables and a bounded slack variable.

References

Anbil, R.; Gelman, E.; Patty, B.; and Tanga, R. 1991, "Recent advances in crew-pairing optimization at American Airlines," *Interfaces*, Vol. 21, No. 1 (January–February), pp. 62–74.

Arntzen, B. C.; Brown, G. G.; Harrison, T. P.; and Trafton, L. L. 1995, "Global supply chain management at Digital Equipment Corporation," *Interfaces*, Vol. 25, No. 1 (January–February), pp. 69–93.

Avery, W.; Brown, G. G.; Rosenkranz, J. A.; and Wood, R. K. 1992, "Optimization of purchase, storage and transmission contracts for natural gas utilities," *Operations Research*, Vol. 40, No. 3 (May–June), pp. 446–462.

Barrodale, I. and Roberts, F. D. K. 1973, "An improved algorithm for discrete l_1 linear approximation," *SIAM Journal on Numerical Analysis*, Vol. 10, No. 5 (October), pp. 839–848.

Bausch, D. O.; Brown, G. G.; and Ronen, D. 1995, "Consolidating and dispatching truck shipments of Mobil heavy petroleum products," *Interfaces*, Vol. 25, No. 2 (March–April), pp. 1–17.

Bowman, E. H. 1963, "Consistency and optimality in decision making," *Management Science*, Vol. 9, No. 2 (January), pp. 310–321.

Bowman, E. H. 1984, "Content analysis of annual reports for corporate strategy and risk," *Interfaces*, Vol. 14, No. 1 (January–February), pp. 61–71.

Brown, G. G.; Clemence, R. D.; Teufert, W. R.; and Wood, R. K. 1991, "An optimization model for modernizing the Army's helicopter fleet," *Interfaces*, Vol. 21, No. 4 (July–August), pp. 39–52.

Brown, G. G.; Cormican, K. J.; Lawphongpanich, S.; and Widdis, D. B. 1997, "Optimizing submarine berthing with persistence incentive," *Naval Research Logistics*, Vol. 44, No. 4 (June).

Brown, G. G.; Dell, R. F.; and Farmer, R. A. 1996, "Scheduling Coast Guard district cutters," *Interfaces*, Vol. 26, No. 2 (March–April), pp. 59–72.

Brown, G. G.; Ellis, C. J.; Graves, G. W.; and Ronen, D. 1987, "Real-time, wide area dispatch of Mobil tank trucks," *Interfaces*, Vol. 17, No. 1 (January–February), pp. 107–126.

Brown, G. G.; Geoffrion, A. M.; and Bradley, G. H. 1981, "Production and sales planning with limited shared tooling at the key operation," *Management Science*, Vol. 27, No. 3 (March), pp. 247–259.

Brown, G. G.; Goodman, C. E.; and Wood, R. K. 1990, "Annual scheduling of Atlantic fleet naval combatants," *Operations Research*, Vol. 38, No. 2 (March–April), pp. 249–259.

Brown, G. G.; Graves, G. W.; and Honczarenko, M. D. 1987, "Design and operation of a multi-commodity production/distribution system using primal goal decomposition," *Management Science*, Vol. 33, No. 11 (November), pp. 1469–1480.

Brown, G. G.; Graves, G. W.; and Ronen, D. 1987, "Scheduling ocean transportation of crude oil," *Management Science*, Vol. 33, No. 3 (March), pp. 335–346.

Brown, G. G. and Olson, M. P. 1996, "A simplex algorithm to maximize persistence and face validity among related model solutions," paper presented at INFORMS National Meeting, Washington, DC (May 5–8).

Charnes, A. and Cooper, W. W. 1961, *Management Models and Industrial Applications of Linear Programming*, John Wiley and Sons, New York.

Charnes, A.; Cooper, W. W.; and Ferguson, R. O. 1955, "Optimal estimation of executive compensation by linear programming," *Management Science*, Vol. 1, No. 2 (January), pp. 138–151.

Dell, R. F. 1997, "Optimizing base realignment and closure implementation," working paper, Naval Postgraduate School, Monterey, California.

Draper, N. R. and Smith, H. 1966, *Applied Regression Analysis*, John Wiley and Sons, New York.

Dundas, J. R. 1996, "Optimal fleet modernization model for the US Navy's combat logistics support helicopter," masters thesis, Naval Postgraduate School, Monterey, California.

Eben-Chaime, M.; Tovey, C. A.; and Ammons, J. S. 1996, "Circuit partitioning via set partitioning and column generation," *Operations Research*, Vol. 44, No. 1 (January–February), pp. 65–76.

Faircloth, J. A. 1989, "An integer programming approach to long range shipbuilding scheduling," masters thesis, Naval Postgraduate School, Monterey, California.

Fletcher, R. 1981, *Practical Methods of Optimization, Volume 2: Constrained Optimization*, John Wiley and Sons, New York.

Fourer, R. 1985, "A simplex algorithm for piecewise-linear programming I: Derivation and proof," *Mathematical Programming*, Vol. 33, No. 2 (November), pp. 204–233.

Fourer, R. 1988, "A simplex algorithm for piecewise-linear programming II: Finiteness, feasibility and degeneracy," *Mathematical Programming*, Vol. 41, No. 3 (September), pp. 281–315.

Fourer, R. 1992, "A simplex algorithm for piecewise-linear programming III: Computational analysis and applications," *Mathematical Programming*, Vol. 53, No. 2 (June), pp. 213–235.

Geoffrion, A. M. and Powers, R. F. 1995, "Twenty years of strategic distribution system design: An evolutionary perspective," *Interfaces*, Vol. 25, No. 5 (September–October), pp. 105–127.

Graves, G. W.; McBride, R. D.; Gershkoff, I.; Anderson, D.; and Mahidhara, D. 1993, "Flight crew scheduling," *Management Science*, Vol. 39, No. 6 (June), pp. 736–745.

Hamming, R. W. 1986, *Coding and Information Theory*, second edition, Prentice-Hall, Englewood Cliffs, New Jersey.

Hurst, E. G. and McNamara, A. B. 1967, "Heuristic scheduling in a woolen mill," *Management Science*, Vol. 14, No. 4 (December), pp. B182–B203.

Huysmans, J. H. B. M. 1970, *The Implementation of Operations Research*, Wiley Interscience, New York.

INSIGHT 1994, *SAILS Concepts: A Handbook for SAILS Users*, Insight, Inc., Alexandria, Virginia.

Jones, C. H. 1967, "Parametric production planning," *Management Science*, Vol. 13, No. 11 (July), pp. 843–866.

Kiewel, K. C. 1985, "Methods of descent for nondifferentiable optimization," *Lecture Notes in Mathematics No. 1133*, Springer-Verlag, Berlin.

Kunreuther, H. 1969, "Extensions of Bowman's theory on managerial decision-making," *Management Science*, Vol. 15, No. 8 (April), pp. B415–B439.

Leachman, R. C.; Benson, R. F.; Liu, C.; and Raar, D. J. 1996, "IMPReSS: An automated production-planning and delivery-quotation system at Harris Corporation—Semiconductor Sector," *Interfaces*, Vol. 26, No. 1 (January–February), pp. 6–37.

Lewandowski, A. and Wierzbicki, A. P. 1989, *Aspiration Based Decision Support Systems*, Springer-Verlag, New York.

Little, J. D. C. 1970, "Models and managers: The concept of a decision calculus," *Management Science*, Vol. 16, No. 8 (April), pp. B466–B485.

Mack, R. P. 1971, *Planning on Uncertainty*, Wiley-Interscience, New York.

March, J. G. and Simon, H. A. 1958, *Organizations*, John Wiley and Sons, New York.

Mifflin, R. 1977, "An algorithm for constrained optimization with semismooth functions," *Mathematics of Operations Research*, Vol. 2, No. 2 (May), pp. 191–207.

Moré, J. J. 1983, "Recent developments in algorithms and software for trust region methods," in *Mathematical Programming—The State of the Art, Bonn 1982*, eds. A. Bachem, M. Grötschel and B. Korte, Springer-Verlag, Berlin, pp. 258–287.

Mulvey, J. M. 1993, "Robust optimization on PC supercomputers," *SIAM News*, Vol. 26, No. 7 (November), pp. 1, 12–14.

Mulvey, J. M.; Vanderbei, R. J.; and Zenios, S. A. 1995, "Robust optimization of large-scale systems," *Operations Research*, Vol. 43, No. 2 (March-April), pp. 264–281.

Ruszczyński, A. 1986, "A regularized decomposition method for minimizing a sum of polyhedral functions," *Mathematical Programming*, Vol. 35, No. 3, (July), pp. 309–333.

Ruszczyński, A. 1993, "Regularized decomposition of stochastic programs: Algorithmic techniques and numerical results," working paper 93–21, (April), International Institute for Applied Systems Analysis, Laxenburg, Austria.

Schrage, L. 1991, *LINDO: An Optimization Modeling System*, fourth edition, Boyd and Fraser, San Francisco, California.

Staniec, C. J. 1996, "Optimizing anti-armor weapons procurement under constrained Department of Defense budgets," paper pre-

sented at INFORMS National Meeting, Washington, DC, May 5–8.

Thuve, H. 1981, "Frequency planning as a set-partitioning problem," *European Journal of Operational Research*, Vol. 6, No. 1 (January), pp. 29–37.

Trull, S. G. 1966, "Some factors involved in determining total decision success," *Management Science*, Vol. 12, No. 6 (February), pp. B270–B280.

Urban, G. L. 1974, "Building models for decision makers," *Interfaces*, Vol. 4, No. 3 (May), pp. 1–11.

Webster's Encyclopedic Unabridged Dictionary of the English Language 1989, Random House, New York.