Progressive Hedging Innovations for a Class of Stochastic Resource Allocation Problems

Jean-Paul Watson,[1] David L. Woodruff,[2] and David R. Strip[3]

Sandia National Laboratories[1]
Discrete Math and Complex Systems Department
P.O. Box 5800, MS 1318
Albuquerque, NM 87185-1318 USA
jwatson@sandia.gov

Graduate School of Management[2]
University of California, Davis
Davis, CA 95616-8609 USA
dlwoodruff@ucdavis.edu

Sandia National Laboratories[3]
Discrete Math and Complex Systems Department
P.O. Box 5800, MS 1316
Albuquerque, NM 87185-1316 USA
drstrip@sandia.gov

Abstract

Progressive hedging (PH) is a scenario-based decomposition technique for solving stochastic programs. While PH has been successfully applied to a number of problems, a variety of issues arise when implementing PH in practice, especially when dealing with very difficult or large-scale mixed-integer problems. In particular, decisions must be made regarding the value of the penalty parameter ρ, criteria for termination, and techniques for accelerating convergence. We investigate these issues in the context of a class of scenario-based resource allocation problem in which decision variables represent resources available at a cost and constraints enforce needs for sufficient combinations of resources. We introduce techniques for selecting ρ in proportion to the unit-cost of the associated decision variable, a mathematically-based heuristic for setting the ρ parameter, novel techniques for convergence acceleration, and methods for detecting and recovering from oscillatory behavior. The efficacy of these techniques is empirically assessed on two test cases: a difficult "laboratory" stochastic network flow problem and a large-scale, real-world, robust spare parts procurement planning problem. Both test problems have integer variables in both stages. Additionally, we demonstrate that variable-specific ρ values are more effective than traditional fixed ρ values, and that the PH algorithm can serve as an effective heuristic even when the mathematical conditions for convergence are not respected.

1 Introduction

When confronted with either a very large or difficult mixed integer stochastic programming problem [10] for which there exist effective techniques for solving individual scenarios, the progressive hedging algorithm proposed by Rockafellar and Wets [17] can serve as an appropriate heuristic. Progressive hedging (PH) is sometimes referred to as a *horizontal* decomposition method because it decomposes stochastic programs by scenarios rather than by time stages. In this paper we introduce innovations that improve the practical performance of PH and report on computational experiments involving a class of stochastic mixed integer programs for resource allocation.

For an individual scenario s, many problems of practical interest can be cast in the general framework of constrained optimization:

$$\begin{array}{ll} \text{minimize} & c \cdot x_s \qquad \text{(SP)} \\ \text{subject to:} & x_s \in \mathcal{Q}_s \end{array}$$

where x_s is a decision vector of length n, c is a cost coefficient vector of length n, and the requirement $x_s \in \mathcal{Q}_s$ expresses the problem constraints, i.e., to ensure x_s is a feasible solution. We use the subscript s to emphasize that the specific problem characteristics will depend on the scenario that is actually observed; the set $\mathcal{S}$ denotes the set of possible scenarios.

Prescient decision makers can simply make use of the decision vector x_s^* that is optimal for the scenario s that they somehow know to be the scenario that will be ultimately realized. All real-world decision makers must make a decision even though *a priori* they are not sure which scenario will be ultimately realized. An optimization model must therefore possess some mechanism(s) for dealing with this uncertainty.

For each scenario $s \in \mathcal{S}$, we denote the probability of occurrence by $\Pr(s)$. These probabilities allow us to take into account prior knowledge of the distribution of individual scenarios, or to weight the relative importance of particular scenarios based on problem-specific knowledge. For the operational decisions discussed in this paper, the general goal is minimize the expected investment cost, which can be written:

$$\begin{array}{lll} \text{minimize} & (c \cdot x) + \sum_{s \in \mathcal{S}} \Pr(s)(f_s \cdot y_s) & \text{(EF)} \\ \text{subject to:} & (x, y_s) \in \mathcal{Q}_s \qquad\qquad \forall s \in \mathcal{S} & \end{array}$$

where the use of the decision vector x (with $x_s = x, \forall s \in \mathcal{S}$) that does not depend on the scenario implicitly implements the *non-anticipativity* constraints that avoid allowing the decisions to depend on the scenario. The y_s represent second-stage, scenario-specific decision vectors with associated cost coefficient vectors f_s, which are determined given x and a particular $s \in \mathcal{S}$.

For such an optimization problem, the basic PH algorithm can be stated as follows, taking a penalty factor $\rho > 0$ and a termination threshold ϵ as the sole input parameters:

1. $k := 0$
2. For all $s \in \mathcal{S}$, $x_s^{(k)} := \text{argmin}_x (c \cdot x + f_s \cdot y_s) : (x, y_s) \in \mathcal{Q}_s$
3. $\bar{x}^{(k)} := \sum_{s \in \mathcal{S}} \Pr(s) x_s^{(k)}$
4. $w_s^{(k)} := \rho(x_s^{(k)} - \bar{x}^{(k)})$
5. $k := k + 1$
6. For all $s \in \mathcal{S}$,

$$x_s^{(k)} := \text{argmin}_x (c \cdot x + w_s^{(k-1)} x + \rho/2 \left\| x - \bar{x}^{(k-1)} \right\|^2 + f_s \cdot y_s) : (x, y_s) \in \mathcal{Q}_s$$

7. $\bar{x}^{(k)} := \sum_{s \in \mathcal{S}} \Pr(s) x_s^{(k)}$
8. For all $s \in \mathcal{S}$, $w_s^{(k)} := w_s^{(k-1)} + \rho \left(x_s^{(k)} - \bar{x}^{(k)} \right)$
9. $g^{(k)} := \sum_{x \in \mathcal{S}} \Pr(s) \left\| x^{(k)} - \bar{x}^{(k)} \right\|$
10. If $g^{(k)} < \epsilon$, then go to step 5. Otherwise, terminate.

Termination criteria other than convergence of the $x_s^{(k)}$ to a common $\bar{x}$ may be necessary due to the fact that non-convergence is a possibility for non-convex optimization problems, as discussed below and further in Section 2.

Integer constraints on elements of the decision vectors x and y_s render stochastic programming problems non-convex and add considerable difficulty to their solution. A variety of algorithms for solving such (mixed-)integer stochastic programming problems have been proposed (e.g., see [13]). For some smaller and comparatively easier problem instances, standard mixed-integer programming (MIP) solvers can be used [16] to directly solve the *extensive form* of the problem in which all scenarios $s \in \mathcal{S}$ are considered simultaneously, as in formulation (EF). However, for the problem instances of interest to us, the extensive forms are either too large or too difficult to solve using standard, commercially available MIP solvers in practical run-times.

In contrast, PH is a natural heuristic algorithm for solving large-scale stochastic mixed integer problems via scenario decomposition. Although the integer variables also add complexity to solution via the PH algorithm, they can be used to speed convergence because equality is well-defined and easily detected [12]. An alternative approach is to use PH to solve a variant of the stochastic program in which the integer constraints are relaxed, and then round to achieve integrality upon termination [11], although for the problems we consider this technique frequently yields poor-quality solutions.

A large class of real-world resource allocation / optimization problems can be characterized by integer variables that represent resources of some sort that can be purchased, with per-unit costs given by the non-negative vector c. For such problems, the binding constraints effectively put lower limits on the values of x, perhaps in complicated ways or perhaps as simple as $Ax \geq b$ for matrix A and vector b with non-negative elements and x constrained to be non-negative. This family of problems is often referred to as diet problems [5], which are sometimes generalized to constrain Ax from both sides. In many diet problems, the decision vector x is only constrained from below. We will refer to this type of problem as a *one-sided* diet problem. The problems we address in this paper resides in this class, and we propose and demonstrate various methods of practically accelerating the convergence of PH for solving this class of problem.

In Section 2 we describe innovations to PH that allow us to compute values for the penalty parameter ρ that is based on the input data, speed convergence, detect non-convergence, and terminate PH based on prospects for further improvement. The computational effectiveness of these techniques is experimentally assessed in Section 3 on both laboratory and real-world instances of the one-sided stochastic resource allocation problem. We then briefly discuss in Section 4 the parallelization of the PH algorithm to yield significant run-time reductions in a deployment environment. We conclude in Section 5 with a discussion of the impact of our results and directions for further research.

2 PH Algorithmic Innovations

Our experience applying PH to large and difficult stochastic mixed-integer programs led us to develop a number of algorithmic enhancements to the basic algorithm, which can be sub-divided into the following three categories: effective ρ value computation (Section 2.1), convergence accelerators (Section 2.2), and termination criteria (Section 2.3). We additionally describe techniques for detecting cycling behavior in Section 2.4.

2.1 Computing Effective ρ Values

We introduce techniques for selecting ρ in proportion to the unit-cost of the associated decision variable and a mathematically-based heuristic for setting the ρ parameter. Variable-dependent ρ strategies have not been previously explored in the literature. Early – and even many recent – experiments concerning PH reported in the literature happen to have used fairly small values of the penalty parameter ρ, with a single scalar used for all variables. For example, Mulvey and Vladimirou [15] report that the best values of ρ were much less than 1 and that performance was sensitive to the choice of ρ. However, the particular value they obtained is an artifact of data scaling. For example, Listes and Dekker [11, p. 374] observe that "there is no conclusive theoretical analysis to support a general selection rule for [ρ]". In their experiments, the best results were obtained using ρ values between 50 and 100.

In the context of the resource allocation problems we consider in Section 3, examination of the weighted objective formula for PH (Step 6 of the pseudocode presented in Section 1) suggests that significantly larger values of ρ may be required to achieve convergence in practical time-frames. We observe that elements $c(i)$ of the cost vector c may range in magnitude from several hundred dollars to several million dollars per decision variable element, which typically represent expensive resources such as vehicles, road routes, spare parts, or other supplies. In the case of an expensive element i, an effective ρ value should be close in magnitude to the unit cost $c(i)$. Otherwise, computation of the initial $w_s^{(k)}(i)$ (Step 4 of the pseudocode in Section 1) will yield a small fraction of $x(i)$ – whose value represents a quantity, commonly less than 100 – and the per-iteration change in the penalty term $w_s^{(k)}(i)x(i)$ will be comparatively small. Slow changes in the penalty terms necessarily yield little movement in $x(i)$, which in turn significantly delays PH convergence. As a corollary, we comment that the optimal ρ value for a given problem need *not* be fixed at a constant value, i.e., the introduction of per-element ρ_i may in fact be more appropriate for some problems, including those examined in Section 3.

Based on these observations, we have developed novel and simple methods for determining element-specific ρ_i values based on problem-specific data. As demonstrated in Section 3, the methods result in substantially improved PH performance relative to constant ρ values, and partially alleviates the need for problem-dependent parameter tuning.

To motivate the method, consider a scalar quantity x for which non-anticipativity must be enforced. Consequently, only a single corresponding w weight multiplier is required. Suppose that x is constrained to be an integer taking on small values. Suppose further, that at optimality $w = w^*$ is quite large. This can occur, for example, when x is the quantity of an expensive resource and other (perhaps numerous) variables represent lower cost operational decisions. If ρ is small, this situation will result in many iterations required for convergence of PH because at each iteration, w can grow only by the product of two small quantities.

Our objective is to develop a heuristic method of setting ρ that will allow the updates to proceed more quickly to a "good" value w^* of the weight w. For practical reasons, we want the magnitude of w to approach from below in order to minimize oscillation or thrashing. Oscillation can occur when the w values are updated too aggressively or converge from both sides particularly in MIPs because the changes in the value of one integer variable can induce changes in others, which are then reversed if the w multiplier "shoots past" its optimal value. Before proceeding, we note that the motivation for our heuristic is based on separability of the decision variables, although it is not required for use of the method. For clarity of the motivation, we proceed in the context of a single variable and linear objective function.

Consider a single decision variable x with corresponding cost coefficient c; individual problem scenarios are denoted by s. After iteration zero of PH completes, we have an estimate of the optimal value for x, which is $\bar{x}^{(0)}$. If we set a value of ρ that will result in $w = c$, then the

proximal term

$$\rho/2 \left\| x_s^{(k-1)} - \bar{x}^{(k-1)} \right\|^2$$

will force the solution to be $\bar{x}^{(0)}$ on the next iteration. The value of w is updated by

$$w_s^{(k)} := w_s^{(k-1)} + \rho \left(x_s^{(k-1)} - \bar{x}^{(k-1)} \right)$$

so the value of ρ for a given scenario s resulting in $w = c$ is

$$\rho_s := \frac{c}{|x_s - \bar{x}^{(0)}|}$$

We want the absolute value of all w elements to approach their ultimate value from below to help mitigate thrashing or cycling that can occur when integers change values forcing jumps in the values of other variables, so we use a bound on the denominator and drop the dependence on s. After PH iteration zero, for each variable x we define $x^{\max} = \max_{s \in \mathcal{S}} x_s^{(0)}$ and $x^{min} = \min_{s \in \mathcal{S}} x_s^{(0)}$. Since $(x^{\max} - x^{\min} + 1) > |x_s - \bar{x}|$ we use

$$\rho(i) := \frac{c(i)}{(x^{\max} - x^{\min} + 1)}$$

for variable i, which does not depend on s. This scheme is used for blending all integer variables in our experiments. In the case of continuous variables can change gradually without jumps, so it is not necessary to use such a conservative denominator. Thus, we instead use the following formula to determine $\rho(i)$ for the continuous variables in our experiments:

$$\rho(i) := \frac{c(i)}{max((\sum_{s \in \mathcal{S}} \Pr(s)|x_s - \bar{x}^{(0)}|), 1)}$$

We denote this heuristic method for selecting per-element $\rho(i)$ by SEP.

The primary advantages of the ρ selection heuristic SEP are its problem-independent nature and the fact that it is parameter-free, eliminating the need for repeated execution of PH in the search for high-quality ρ values. However, there exists a high likelihood that more effective methods exist for any specific problem. We have investigated a number of alternative ρ selection strategies for a range of stochastic mixed-integer programs. The best-performing alternatives (including SEP) were all based on the simple observation that the value of $\rho(i)$ should be proportional to element unit cost, as discussed above. We also report on results based on a straightforward yet effective "cost-proportional" method for setting $\rho(i)$. Specifically, we set $\rho(i)$ equal to a multiplier $k > 0$ of the element unit cost $c(i)$. The method is denoted by CP$(\cdot)$, where CP stands for *cost-proportional* and the argument gives the cost multiplier. Finally, as a control measure, we consider the performance of PH using various fixed, global values of ρ. We denote the corresponding method by FX$(\cdot)$, where the FX stands for *fixed* and the argument gives the value of ρ.

2.2 Accelerating Convergence

Although PH may eventually drive agreement among the decision variable vectors x_s to a common vector x, in practice the number of iterations required is frequently excessive for complex, non-convex stochastic integer programming problems.

The following three acceleration methods are designed for one-sided constraints, such as when the problem for each scenario is to minimize $c \cdot x$ subject to $Ax \geq b$ with $x \geq 0$ where the elements of vectors c and b and the matrix A are all non-negative. For problems where the

constraints effectively limit x from both sides, these methods may result in PH encountering infeasible scenario sub-problems even though the problem is ultimately feasible. For one-sided problems, as we will demonstrate, the methods are however quite effective.

A detailed analysis of PH algorithm behavior on the problems considered in Section 3 in addition to other problems indicates that individual decision variables $x_s(i)$ frequently converge to specific, fixed values z for all $s \in \mathcal{S}$ in early PH iterations. Further, despite interactions among the $x_s(i)$ for any particular scenario s, the value of z frequently fails to change in subsequent PH iterations. Such variable "fixing" behaviors lead to a potentially powerful, albeit obvious, heuristic: once $x_s^{(k)}(i) = x(i)$ for all $s \in \mathcal{S}$ at a particular PH iteration k, fix $x_s^{(l)}(i) = z$ for all subsequent iterations $l > k$. For continuous decision variables, a small differential threshold must clearly be imposed. As shown in Section 3, variable fixing can yield substantial reductions in solution times by accelerating (through variable elimination) the solution times for individual scenario sub-problems, at the expense of slight reductions in solution quality.

In applying this heuristic, we first introduce a *lag* parameter $\mu \in \{0, 1, ...\}$. Consider a given PH iteration k. We fix $x_s^{(k)}(i)$ for all subsequent iterations $l > k$ once $x_s^{(m)}(i) = z$ for all $s \in \mathcal{S}$ and $m \in \{k-\mu|\mathcal{S}|, \ldots, k\}$, such that $m \geq \mu|\mathcal{S}|$. In other words, we fix decision variables once their value has stabilized to a fixed z over the last $\mu|\mathcal{S}|$ PH iterations. Low values of μ yield immediate or near-immediate variable fixing; larger values of μ can respond to the empirically rare event that the value of z may in fact vary over moderate time horizons, i.e., it may become "undone" due to the influence of competing decision variables. The multiplicative factor $|\mathcal{S}|$ accounts for the observation that the number of PH iterations required for convergence grows with the total number of scenarios under consideration.

This idea can be taken further by fixing values for decision variables that have not yet converged as a means of quickly forcing termination of the algorithm, which we refer to as *slamming*. Consider a situation in which it has been determined that the individual scenario solutions $x_s^{(k)}$ are "sufficiently" converged, i.e., they are very nearly homogeneous in both the values of the decision vectors $x_s^{(k)}$ and the scenario costs $c \cdot x_s^{(k)}$. The basic PH algorithm can take very large number of iterations to resolve the remaining discrepancies, despite minimal impact on the final solution quality. One alternative, reported in the literature [11, 12], is to solve a variant of the extensive form in which all currently-converged decision variables are fixed to their common value. Another alternative, explored here, is to force absolute PH convergence via aggressive variable fixing.

Once the variables have converged sufficiently (the specific criteria are described subsequently in Section 2.3), we first set the lag $\mu = 0$, independent of its current value. Then, every other iteration we identify the free decision variable $x(i)$ for which the total cost $c(i) \cdot \max_{s \in \mathcal{S}} x_s(i)$ is minimal. We then fix x1$x(i) = \max_{s \in \mathcal{S}} x_s(i)$. Given our one-sided formulation, feasibility of the scenario sub-problems is necessarily not lost via such a maximum-value scheme. Clearly, this scheme is guaranteed to force termination. We have investigated performance using various alternative intervals, but performance is not sensitive to this choice.

2.3 Termination Criteria

In practice, PH empirically yields large reductions in $|x_{s_1} - x_{s_2}|$ for $s_1, s_2 \in \mathcal{S}$ in early iterations, while the remaining and majority of iterations serve in a fine-tuning role to drive the already small differences in $|x_{s_1} - x_{s_2}|$ to 0. To detect near-convergence in the solution vectors x_s ($s \in \mathcal{S}$) we first define the normalized average per-scenario deviation from the "average" solution *td* $= (\sum_{i,s:\overline{x}(i)>0} \frac{|x_s(i)-\overline{x}(i)|}{\overline{x}(i)})/|\mathcal{S}|$, where $\overline{x}(i)$ represents the average of $x_s(i)$ over all $s \in \mathcal{S}$. We then can invoke variable slamming to quickly force PH convergence once *td* drops below some parametric threshold λ_t. The value of λ_t places a threshold on the degree of heterogeneity allowed in the set of solutions x_s.

Such a termination criterion assumes that small differences in $|x_{s_1} - x_{s_2}|$ are correlated with small differences in the costs $|c \cdot x_{s_1} - c \cdot x_{s_2}|$. In practice, however, this is often *not* the case. For example, there can exist "holdout" scenarios that require more high-cost resources than other scenarios to achieve feasibility. Consequently, the value of *td* may in fact be very small, while the discrepancy in overall costs may be quite large. To protect against such situations, we additionally consider a termination criterion based on the variability of solution quality in any given PH iteration k. For this problem, upper bounds on the $x(i)$ are easily obtained, e.g., by considering the total number of a resource i that could ever be used in a given scenario $s \in \mathcal{S}$. At an arbitrary PH iteration k, consider the solutions x_s for all scenarios $s \in \mathcal{S}$. Let x^{max} denote the decision vector whose elements represent the maximal value appearing in any solution x_s, i.e., $x^{max}(i) = max_{s \in \mathcal{S}}(x_s^{(k)}(i))$. The element-wise minimum vector x^{min} is defined analogously. Clearly, x^{max} is a feasible (albeit likely suboptimal) solution to all scenarios $s \in \mathcal{S}$, while x^{min} is generally infeasible. Finally, let $qd = (c \cdot x^{max} / c \cdot x^{min}) * 100$. We can then terminate PH iterations once *qd* drops below a parameterized threshold value λ_q, e.g., where $\lambda_q = 1\%$.

In our PH implementations, we invoke variable slamming once *both* $td \leq \lambda_t$ and $qd \leq \lambda_q$ after a PH iteration k.

2.4 Detecting Cyclic Behavior

Finally, we note that for all types of non-convex stochastic programs, there is a risk of non-convergence of the PH algorithm. In the experiments discussed in Section 3, cycling is detected at least once in roughly one tenth of all algorithmic trials. To detect cycles, we chose to focus on repeated occurrences of $w_s(i)$ vectors, implemented using a simple hashing scheme [26] to minimize impact on run-time. Once a cycle is detected for any decision variable $x(i)$, the value of $x(i)$ is immediately fixed to $\max_{s \in \mathcal{S}} x_s(i)$; feasibility is again ensured in the case of one-sided resource constraints. In practice, few variables are fixed in such a fashion, yielding minimal impact on final solution quality while assuring termination.

3 Experimental Analysis of PH Performance

We now describe an empirical performance analysis of the various PH algorithmic techniques proposed in Section 2. We begin in Section 3.1 with a simple, yet computationally difficult, two-stage stochastic network flow problem. Here, the extensive form of the problem is small enough to be stored in RAM, but too difficult for commercial MIP solvers to solve in tractable run-times. Next, in Section 3.2, we consider a real-world spare parts procurement planning problem, formulated as a two-stage, robust integer program. In this case, *both* the extensive form and individual scenarios are too difficult and too memory-intensive for solution via commercial MIP solvers. Consequently, heuristics are used to generate solutions to individual scenarios, with PH serving as the mechanism to enforce non-anticipativity.

3.1 A Stochastic Network Flow Problem

To provide an easily reproduced laboratory example for demonstrating and testing our proposed methods, we first consider a basic network flow model with stochastic and resource allocation aspects. The problem formulation is provided in Section 3.1.1. Our problem instance generation scheme is discussed in Section 3.1.2, in addition to our experimental methodology. Computational results are then detailed in Section 3.1.3.

3.1.1 Stochastic Programming Formulation

A flow network is defined over a node set $\mathcal{V}$ and an arc set $\mathcal{A}$, $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$. The network structure is static across a set of scenarios $\mathcal{S}$. For each scenario $s \in \mathcal{S}$, a quantity $D_{kl}(s)$ must be shipped between each pair of nodes $(k, l) \in \mathcal{V} \times \mathcal{V}$, $k \neq l$. The first-stage decision variables includes continuous arc capacities $x(a)$, $a \in \mathcal{A}$, and binary indicators of arc availability $b^0(a)$, for which the cost parameters $c(a)$ and $F^0(a)$ are respectively defined. Additional resources for each arc $a \in \mathcal{A}$ must be deployed after the demand becomes known in order to activate the arc. The use of these resources is defined by the second-stage decision variables $b(a, s)$ at a corresponding cost of $F(a, s)$, introduced for each $a \in \mathcal{A}$ and $s \in \mathcal{S}$. The optimization objective is then to minimize the sum of the first stage fixed plus marginal costs and the expected second-stage costs.

Continuous variables $y_{kl}(a, s)$ are introduced to represent the flow from node k to node l passing through arc $a \in \mathcal{A}$ in scenario $s \in \mathcal{S}$. Assuming the availability of a large constant M (e.g., one plus the sum of all demands), the problem formulation, which we denote (SNF), is given as:

$$\text{minimize} \quad \sum_{a \in \mathcal{A}} \left[c(a)x(a) + F^0(a)b^0(a) + \sum_{s \in \mathcal{S}} \Pr(s) F(a, s) b(a, s) \right] \tag{SNF}$$

$$\text{subject to:} \quad \sum_{a \in \mathcal{A}^+(\nu)} y_{kl}(a, s) - \sum_{a \in \mathcal{A}^-(\nu)} y_{kl}(a, s) = \begin{cases} -D_{kl}(s) & \text{if } \nu = k \\ D_{kl}(s) & \text{if } \nu = l \\ 0 & \text{otherwise,} \end{cases}$$

$$\forall \nu, k, l \in \mathcal{V}, s \in \mathcal{S}$$

$$x(a) \geq \sum_{k,l \in \mathcal{V}} y_{kl}(a, s) \qquad \forall a \in \mathcal{A},\ s \in \mathcal{S}$$

$$y_{kl}(a, s)) \leq M b^0(a) \qquad \forall k, l \in \mathcal{V}, a \in \mathcal{A}$$

$$y_{kl}(a, s) \leq M b(a, s) \qquad \forall k, l \in \mathcal{V}, a \in \mathcal{A}, s \in \mathcal{S}$$

$$b^0(a) \in \{0, 1\} \qquad \forall a \in \mathcal{A}$$

$$b(a, s) \in \{0, 1\} \qquad \forall a \in \mathcal{A}, s \in \mathcal{S}$$

$$y_{kl} \geq 0 \qquad \forall k, l \in \mathcal{V}, k \neq l$$

$$x(a) \geq 0 \qquad \forall a \in \mathcal{A}$$

where the notations $\mathcal{A}^+(\nu)$ and $\mathcal{A}^-(\nu)$ respectively indicate the set of arcs into and out of node $\nu \in \mathcal{V}$. The probability of $s \in \mathcal{S}$ being realized is denoted by $\Pr(s)$. The (SNF) formulation represents the extensive form of the optimization problem; the corresponding single-scenario quadratic MIPs, for use with PH, are straightforwardly derived.

3.1.2 Test Problems and Experimental Methodology

The computational experiments reported in Section 3.1.3 were performed using networks consisting of 10 nodes, consecutively labeled from 1 to 10. The static arc set is given by the following node pairs, each representing a bi-directional arc: (1,2), (1,3), (2,3), (2,4), (2,5), (3,4), (4,5), (5,6), (6,7), (6,8), (7,8), (7,9), (7,10), (8,9), and (9,10).

To shorten the exposition and ease reproduction of our results, we use simple, arbitrary formulas to establish parameter values for the instance data. For each arc $a \in \mathcal{A}$ connecting nodes indexed by i and j, we let $c(a) = 100 + 10|i - j| + (ij \mod 10)$. As in [18], the $D_{kl}(s)$ for a specific scenario $s \in \mathcal{S}$ are given by $D_{kl}(s) = 0.1D(s) + \epsilon_{kl}$, where $D(s)$ represents the aggregate flow volume for scenario s, sampled from a normal distribution $\mathcal{N}(30, 5)$; the ϵ_{kl} are independent normally distributed variables with mean 0 and standard deviation 0.25. We let $F^0(a) = 10c(a)$ and $F(a, s) = U(s)c(a)$ where $U(s)$ is drawn from a distribution that assigns equal likelihood to the integers 5, 10, and 15. We additionally require all arcs to be bi-directional and the capacities are assumed to be symmetric across an arc, i.e., $a_1 = (i, j) \wedge a_2 = (j, i) \Rightarrow (x(a_1) = x(a_2) \wedge b^0(a_1) = b^0(a_2))$. For testing purposes, we generated 10 and 50-scenario problem sets, each containing 5 instances apiece with $\Pr(s) = 1/|\mathcal{S}|$. While small in

terms of both network scale and $|\mathcal{S}|$, these instances pose significant computational challenges to both our PH variants and commercial MIP solvers, due to the presence of integer variables in both the first and second decision stages.

To establish a performance baseline and to bound final solution quality, we allocated two days (2880 minutes) of run-time to ILOG's [8] CPLEX 10.1 mixed-integer programming solver to the extensive form (SNF) of each problem instance, with the best feasible incumbent solution recorded upon termination. A primary experimental objective is to demonstrate the relative benefit of variable versus fixed ρ selection strategies. Consequently, we consider the following ρ selection strategies for PH: CP(1.0), SEP, FX(250), and FX(1000). The parameters for the two fixed ρ strategies – 250 and 1000 – are respectively chosen based on the approximate mean values of the (SNF) cost parameters $c(a)$ and $F^0(a)$. A secondary experimental objective is to assess the impact of the fix lag parameter μ on PH performance, which we vary among $\{0, 1, 2\}$. To additionally examine interaction effects between the ρ selection strategies and the fix lag μ, we performed a fully crossed experiment, yielding 12 runs for each instance.

For each individual PH run, we set the termination criteria parameters λ_q and λ_t equal to $.01\%$ and 0.0001, respectively. Individual PH scenarios are also solved with CPLEX 10.1, leveraging the quadratic solver engine. To accelerate scenario solves in early PH iterations [25] – which are empirically far more costly than those for later iterations – we monotonically non-increase the CPLEX *mipgap* parameter as PH progresses, setting the value equal to the minimum of the convergence parameter *td* computed at the end of the previous PH iteration and the *mipgap* parameter value used in the previous PH iteration. With the exceptions noted below, all PH variants are executed until convergence to a common x is achieved through the use of variable slamming or other mechanisms described in Section 2. At each PH iteration, a maximal-cost solution satisfying all scenarios can be formed by taking the element-wise maximum across the decision variables for all $s \in \mathcal{S}$; feasibility is guaranteed due to the one-sided nature of the resource constraints, and the cost is given as:

$$\sum_{a \in \mathcal{A}} \left[c(a)x^{\max}(a) + F^0(a)b^{0\,\max}(a) + \sum_{s \in \mathcal{S}} \Pr(s)F(a,s)b(a,s) \right] \tag{1}$$

The best solution found in any PH iteration is recorded, in addition to the overall run-time, rounded to the nearest half-minute increment. All runs are executed on a 64-bit AMD Opteron 2.2GHz workstation, with 64GB of RAM running Linux 2.6.

3.1.3 Results

We first consider the results for the 10-scenario instances. Despite their small size, the extensive forms of these instances are difficult for CPLEX 10.1. In no case could CPLEX prove optimality within the two day limit, instead yielding a final optimality gap between 1.32% and 2.15%; improvements in the incumbent were observed throughout the run. To assess the performance of PH, we compute the percentage improvement of each algorithm over the best CPLEX incubment; negative numbers indicate PH was outperformed by CPLEX. The average results across the 5 instances are reported in Table 1, in addition to the average run-time and number of PH iterations.

Analyzing the PH results, we observe that the fixed ρ strategies significantly underperform the variable ρ strategies, controlling for the fix lag μ. Runs with FX(1000) yielded the worst performance in terms of solution quality, which is consistent with the overweighting of ρ values (relative to the mean $c(a)$) associated with arc capacities $x(a)$. Intuitively, we expect lower ρ values to yield improved solution quality at the expense of increased run-times. This is observed in runs with FX(250). However, the FX(250) strategy consistently underperforms both the CP(1.0) and SEP variable ρ strategies in terms of solution quality. Further, the CP(1.0) requires

PH Algorithm	Improvement Versus Baseline	Average Run-Time	Average PH Iterations
$\mu = 0$, FX(1000)	-14.00	7.6	20.4
$\mu = 0$, FX(250)	-4.75	17.5	63.8
$\mu = 0$, CP(1.0)	-1.81	9.7	41
$\mu = 0$, SEP	**-1.27**	18.9	86.6
$\mu = 1$, FX(1000)	-10.87	75.6	23.4
$\mu = 1$, FX(250)	-2.848	85.7	117
$\mu = 1$, CP(1.0)	-0.72	51.5	61.6
$\mu = 1$, SEP	**-0.68**	61.6	152.8
$\mu = 2$, FX(1000)	-10.21	141.4	23.6
$\mu = 2$, FX(250)	-1.57	129.4	96
$\mu = 2$, CP(1.0)	**-0.73**	73.005	68
$\mu = 2$, SEP	-0.85	321.8	321.8

Table 1: Performance results for PH runs on the five 10-scenario stochastic network flow instances. Performance is relative to the best solution found by CPLEX 10.1 for the extensive form within a time limit of 2800 minutes, and is averaged over the 5 instances. Average run-times are reported in minutes, rounded to the nearest half-minute increment. Given a fixed μ, the PH configuration yielding the highest-quality performance relative to CPLEX baseline is bold-faced.

PH Algorithm	Improvement Versus Baseline	Average Run-Time	Average PH Iterations
$\mu = 0$, FX(250)	6.06	305	186
$\mu = 0$, FX(1000)	-3.58	66	163
$\mu = 0$, CP(1.0)	8.16	274	99
$\mu = 0$, SEP	6.92	1337	485

Table 2: Performance results for PH runs on the five 50-scenario stochastic network flow instances. Performance is relative to the best solution found by CPLEX 10.1 for the extensive form within a time limit of 2800 minutes, and is averaged over the 5 instances. Average run-times are reported in minutes, rounded to the nearest half-minute increment. Given a fixed μ, the PH configuration yielding the highest-quality performance relative to CPLEX baseline is bold-faced.

less run-time in all cases. The mathematically motivated SEP variable ρ strategy outperforms the CP(1.0), except when $\mu = 2$. Independent of the ρ selection strategy, increases in μ marginally improve final solution quality, but at the expense of significant growth in run-times. While we cannot rule out fixed values of ρ that may outperform our variable ρ strategies, we observe that no tuning was performed for the variable ρ strategies (which would mitigate the run-time advantage of fixed ρ strategies), and that the fixed ρ values selected were reasonable and based on instance data.

While none of the variable ρ PH configurations outperform the CPLEX baseline in terms of final solution quality (on average; a SEP PH configuration identified a better solution on a single instance), this was expected given the small instance size. Further, the run-times are significantly less than the 2800 minutes allocated to CPLEX. Even though the instances are relatively small, the variable ρ strategies outperform CPLEX on the extensive form of (SNF) given equivalent run-times.

We next consider results for the five 50-scenario instances, reported in Table 2. Based on our 10-scenario results, we limit the scope of the experiments. PH run-time empirically

grows superlinearly with increases in $|\mathcal{S}|$ and μ, necessitating a more limited investigation. Consequently, we limit $\mu = 0$ due to the growth in run-times, caused by the total number of PH iterations and the aggregate CPLEX run-times on individual scenarios. These instances are significantly more difficult than the 10-scenario instances, with CPLEX optimality gaps on the extensive form ranging between 15.85% and 18.82% upon termination.

All but the FX(1000) PH configurations outperform the CPLEX extensive form baseline in terms of both run-time and solution quality, with the best performance obtained by CP(1.0). The latter yields solutions over 8% better than the CPLEX baseline in an order-of-magnitude less run-time. Both variable ρ strategies outperform the fixed FX(250) strategy, with the CP(1.0) variant obtaining higher-quality solutions in less run-time. The performance of the FX(250) variant is, however, reasonably competitive with the variable ρ configurations. It is not surprising that specific, fixed ρ values can yield competitive performance on these laboratory instances, given relatively homogeneous resource costs. For real-world problems where resource costs are significantly more variable (as shown in Section 3.2) SEP and other variable ρ selection strategies provide more distinct advantages. The parameter-free SEP strategy obtains high-quality solutions, but with longer run-times relative to CP(1.0). However, this advantage is at least partially mitigated by the need to select reasonable parameter values for the CP$(\cdot)$ approach. Finally we observe that the various convergence acceleration and cycle detection techniques described in Section 2, in particular fixing the lag μ to 0, are required to achieve the reported performance levels.

3.2 A Spare Parts Support Enterprise Problem

We next assess our proposed methods in the context of a very large-scale resource allocation problem involving procurement planning for a spare parts distribution and repair network for aviation fleets. Introductory background on this important practical problem is provided in Section 3.2.1.

A robust integer programming formulation of this problem and heuristics for solving individual scenarios are presented in Section 3.2.2. We refer to this problem formulation as *robust* as opposed to *stochastic* because the $\sum_{s\in\mathcal{S}} \Pr(s)(f_s \cdot y_s)$ term in the (EF) formulation equals 0 due to $f_s = 0$ for all $s \in \mathcal{S}$, i.e., there are no second-stage costs in the variant we consider. Ideally, it would be possible to solve the extensive form of the robust formulation directly with existing commercial MIP solvers. However, this is not currently feasible in part due to the size of the resulting MIPs for even relatively small instances, exceeding the memory limits of most modern workstations. Thus, scenario decomposition is essential because some industrial problem instances are much larger than those considered here. Additionally, the solve times for even the LP relaxation of the extensive form can extend to several days on powerful 64-bit workstations; MIP solves are therefore not practical in the foreseeable future.

The test problems and experimental methodology are respectively discussed in Sections 3.2.3 and 3.2.4. Computational results on two problem types are then detailed in Sections 3.2.5 and 3.2.6.

3.2.1 Background and Terminology

Maintaining deployed aircraft to sustain operations at low cost is a central problem in aviation fleet management [14]. For military fleets, the operations requirement is typically codified by requiring that some minimal proportion of aircraft in each deployed unit (e.g., squadron) are always available to fly. This proportion, referred to as *operational availability*, is driven at a unit level by two primary factors: availability of resources to perform aircraft maintenance and spare parts to replace failed aircraft components. At the enterprise level, spare parts availability to

supply demands is influenced by the capacity of repair depots to fix failed parts, the initial stockage and re-order policies of supply depots, and replacement part procurement lead times. Fleet sustainability cost is then dictated by the costs associated with initial spare parts stock procurement and the resources allocated to repair and maintenance activities. Various secondary costs, including those associated with the one-for-one replacement of consumable parts, site-to-site transportation, and shop materials for part repair, are generally ignored; such costs are "sunk" in the sense that they are in practice unavoidable once the spares and resource levels throughout a system are determined. This basic sustainability problem extends beyond aircraft fleets, to systems including military ground combat brigades, oil rigs, computing infrastructures, and commercial trucking companies.

A range of analytic optimization models for sustainability problems – with an emphasis on military aircraft fleets – has been developed over the last 40 years, beginning with METRIC [20] and more recently with ASM [23]. While useful, these models generally embody a large set of assumptions, e.g., fixed repair turn-around times and specific parametric part failure distributions. To more accurately represent real-world sustainability systems, many military and commercial entities have recently turned to simulation models. One example of relevance here is the Support Enterprise Model (SEM) [24], a complex, highly detailed discrete event simulation jointly developed by Lockheed Martin and Sandia National Laboratories for use on the Lockheed Martin Joint Strike Fighter (JSF) program. While eliminating assumptions present in the METRIC-like models and facilitating the consideration of complex business rules, simulation models such as SEM lack an inherent optimization capability. Thus, novel optimization models must be developed.

In any given SEM problem instance, the decision variables consist of (1) stock re-order levels for each part at each site in the system; relevant sites include all supply depots (independent of echelon level), OEMs (Original Equipment Manufacturers), and bases, and (2) assigned quantities for all repair-related resources for each site in the system; relevant sites include repair depots, OEMs, and bases. Resources associated with aircraft maintenance, e.g., technicians or engine lifts, are determined by an external analysis tool, and are considered fixed in the enterprise optimization. Although not required by SEM, we assume an $(s, s-1)$ inventory re-order policy, such that the initial procurement level for each part/site combination is identical to the re-order level. Resource-related costs include both initial procurement (if any) and on-going operational and maintenance costs. The elements of the cost vector c are then per-item, per-site procurement and operational costs. We denote this general class of optimization problem as Support Enterprise Sustainability Problems (SESPs).

Given a problem instance, we use the SEM simulator to generate part failure data at each base in the system for $|\mathcal{S}|$ independent replications or scenarios. These replications are executed in a "flooded" mode, i.e., in which the supply of parts and resources is unconstrained. By directly leveraging SEM in this manner, it is possible to sample part failures from non-parametric distributions arising from, for example, complex wear-out patterns or combat damage. The resulting part failure sequences are optimistic relative to a cost-constrained environment. In particular, part failures are assumed to be independent. For example, consider a part failure sequence for a particular plane from a flooded SEM replication in which a landing gear component fails on day n and the engine fails on day $n+5$. In a resource-rich environment, the landing gear is quickly repaired, such that the engine will fail due to the aircraft being operational; in a resource-constrained environment, lack of a spare landing gear component may down the plane for $n > 5$ days, in which case the engine failure would be delayed. Given the typically high availability requirements (all but a handful of planes at a base) for JSF and other aviation fleets, the degree of conservatism is in practice not significant.

We next describe a robust optimization model for the SESP, based on input data derived from the SEM simulation. SEM is not, however, used to evaluate constraints or cost functions. While there is a rich literature on this topic (simulation-based optimization) [7], "black-box"

approaches are not feasible for our problem; individual replications require minutes to hours of run-time, and the number of decision variables is very large. Although not discussed further in this paper, simulators such as SEM are in practice used as a final verification step, i.e., the ultimate quality of any solution is assessed via discrete-event simulation. In summary, high-fidelity simulation provides the input for and final validation of our solutions, but is otherwise not used in the process of generating those solutions.

3.2.2 Integer Programming Formulation and Heuristic Solution Alternatives

We now describe a robust integer programming formulation of the SESP extensive form. Due to the complexity of the full formulation, we defer to [6] for a complete description. Here, we ignore some of the more intricate, problem-specific constraints, and instead focus on a complete description of a slightly simplified model. The full model, expressed in the AMPL mathematical programming language [2], is available from the authors.

Each instance of the SESP specifies a set $\mathcal{B}$ of bases, $\mathcal{SD}$ of supply depots, $\mathcal{RD}$ of repair depots, and $\mathcal{O}$ of OEMs. The set $\mathcal{A} = \mathcal{B} \cup \mathcal{SD} \cup \mathcal{RD} \cup \mathcal{O}$ denotes the aggregate set of sites, with the intent of creating a unified site index set. We coarse-grain time in the formulation to a daily resolution (with appropriate rounding to ensure conservativeness), with the time index t ranging from 1 to a maximal value T.

The set of part types in the system is denoted by $\mathcal{P}$. The cost of a single instance of a part $p \in \mathcal{P}$ is denoted $c[p]$. A site $a \in \mathcal{A}$ stocks a set of part types $p[a] \subseteq \mathcal{P}$. The number of days required to transport a part $p \in \mathcal{P}$ between two sites $a_1, a_2 \in \mathcal{A}$ is denoted $st[p, a_1, a_2] \geq 1$. A base $b \in \mathcal{B}$ receives resupply of a part p from a supply depot $supplier[p, b] \in \mathcal{SD}$; a supply depot $sd \in \mathcal{SD}$ receives resupply of a part p from an OEM $supplier[p, sd] \in \mathcal{O}$. Parts can fail in a number of distinct modes or levels $l \in \mathcal{L}$. The set of failure modes possible for a given part p is denoted $fm[p] \subseteq \mathcal{L}$. OEMs are assumed to be uncapacitated in this simplified formulation, with a build lead time $bt[p, o]$ for a part $p \in \mathcal{P}$ at OEM $o \in \mathcal{O}$.

The set of resource types in the system is denoted by $\mathcal{R}$. The per-unit cost of a resource $r \in \mathcal{R}$ is denoted $c[r]$. A site $a \in \mathcal{A}$ possesses the set of resource types $r[a] \subseteq \mathcal{R}$ required to accomplish the possible repairs at that site. The quantity of a resource $r \in \mathcal{R}$ required to repair a part $p \in \mathcal{P}$ at failure mode $l \in \mathcal{L}$, independent of site, is denoted by $rq[p, l, r] \geq 0$.

The failure data from the SEM simulator yields, on a daily basis, the number of failures of part $p \in \mathcal{P}$ at a base $b \in \mathcal{B}$ with failure mode $l \in \mathcal{L}$. Each individual count represents the failure of a part on a plane flying a mission on day t. A total of $|\mathcal{S}|$ distinct scenarios are generated. The corresponding quantities are denoted $pf[p, l, b, t, s]$, where $t \in [1..T]$ and $s \in \mathcal{S}$.

Upon failure, a *consumable* part (e.g., a tire) is immediately disposed of, while all other parts are shipped to the appropriate site for repair. Consumable parts are represented by a failure mode $l = -1$; all other failure modes indicate a repairable part. From a base $b \in \mathcal{B}$, it is necessary to determine the site in the logistics chain capable of repairing the failed part. This logic is ultimately encoded in a look-up table, with $repairer[p, l, b]$ denoting the target site for a part $p \in \mathcal{P}$ with failure mode $l \in \mathcal{L}$ at base $b \in \mathcal{B}$. Note that $repairer[p, l, b] = b$ in the case of base-repairable parts; otherwise, $repairer[p, l, b] \in \mathcal{RD}$. Once a carcass arrives at a repair site, it immediately enters the repair queue. Once resources are available, the repair process is initiated; the time required to execute a repair task for part p with failure mode l is denoted $rt[p, l]$. Parts repaired at a base immediately re-enter inventory at that base upon completion of the associated repair task; parts repaired at a repair depot $rd \in \mathcal{RD}$ enter inventory after shipment to a supply site $repairdest[p, rd] \in \mathcal{SD}$.

Because OEMs are non-capacitated in the simplified model, and we assume a one-for-one replenishment policy and unit batch sizes, part builds are initiated at time t for each occurrence of a part failure at a base b at time t. Let $oemso[p, o, sd, t, s]$ denote the number of parts of type $p \in \mathcal{P}$ shipped from an OEM $o \in \mathcal{O}$ to a supply depot $sd \in \mathcal{SD}$ at time t in scenario s, such

that $p \in p[\mathit{sd}]$, $p \in p[o]$, $\mathit{supplier}[p, \mathit{sd}] = o$, and $-1 \in \mathit{fm}[p]$. We then define:

$$\mathit{oemso}[p, o, \mathit{sd}, t, s] = \sum_{b \in \mathcal{B}: \mathit{supplier}[p,b] = \mathit{sd}} \mathit{pf}[p, -1, b, \mathit{max}(0, t - \mathit{bt}[p, o], s] \tag{2}$$

where we assume $\mathit{pf}[p, l, b, 0, s] = 0$. Additionally, also for notational convenience, we assume $\mathit{pf}[p, l, b, t, s] = 0$ for all instances in which $l \neq -1$.

For each base $b \in \mathcal{B}$, there are $\mathit{ad}[b]$ possible *air days* available for flying missions over the time horizon $[1..T]$, i.e., the product of the number of planes available at base b and T. The performance criterion common across all SESP scenarios is then expressed simply as the maintenance of a minimal fraction of available air days $\nu[b]$ ($0 \leq \nu \leq 1$) of $\mathit{ad}[b]$, for all $b \in \mathcal{B}$.

The first-stage decision variables in the SESP robust formulation are the inventory and resource levels at each site in the system. The set of part/site and resource/site pairs are respectively denoted by $\mathcal{PV} \subseteq \mathcal{P} \times \mathcal{A}$ and $\mathcal{RV} \subseteq \mathcal{R} \times \mathcal{A}$. The corresponding decision variables are denoted by $\mathit{pl}[p, a]$ and $\mathit{rl}[r, a]$, with $(p, a) \in \mathcal{PV} \wedge p \in p[a]$ and $(r, a) \in \mathcal{RV} \wedge r[a]$. All SESP second-stage decision variables involve the tracking of part- and resource-related statistics at each site through time. For bases, we track the number of parts on-hand, due in from a supplier, and due out to a downed aircraft; these are respectively denoted $\mathit{boh}[p, b, t, s]$, $\mathit{bdi}[p, b, t, s]$, and $\mathit{bdo}[p, b, t, s]$ for all $t \in [1..T]$, $s \in \mathcal{S}$, and $(p, b) \in \mathcal{PV}$. Supply depots track (1) the number of parts on hand $\mathit{sdoh}[p, \mathit{sd}, t, s]$ for all $t \in [1..T]$, $s \in \mathcal{S}$, and $(p, sd) \in \mathcal{PV}$, and (2) the number of parts shipped to a base $\mathit{sdso}[p, b, \mathit{sd}, t, s]$ for all $t \in [1..T]$, $s \in \mathcal{S}$, $(p, sd) \in \mathcal{PV}$, and $b \in \mathcal{B}$. Finally, repair depots and bases track the following quantities for all (p, l) pairs that can be repaired at the corresponding site: (1) the number of parts awaiting repair $\mathit{rdwr}[p, l, \mathit{rd}, t, s]$ for all $t \in [1..T]$, $s \in \mathcal{S}$, $\mathit{rd} \in \mathcal{RD}$, and $l \in \mathit{fm}[p]$, (2) the analogous quantity $\mathit{rdbr}[p, l, \mathit{rd}, t, s]$ representing the number of parts initiating repair at time t in scenario s, and (3) the number of repaired parts $\mathit{rdso}[p, \mathit{rd}, \mathit{sd}, t, s]$ of type $p \in \mathcal{P}$ shipped from each repair depot $\mathit{rd} \in \mathcal{RD}$ to a supply depot $\mathit{sd} \in \mathcal{SD}$ at time step t in scenario s. All first and second-stage variables are required to be integral, with values ≥ 0.

We assume in the simplified SESP formulation a standard $(x, x - 1)$ inventory re-order policy, i.e., one-for-one replenishment. Under this policy, we enforce inventory "conservation" for each part at each base via the following constraints, instantiated for all $s \in \mathcal{S}$ and $(p, b) \in \mathcal{PV} : b \in \mathcal{B}$:

$$\mathit{boh}[p, b, 1, s] - \mathit{bdo}[p, b, 1, s] = \mathit{pl}[p, b] - \sum_{l \in \mathit{fm}[p]} \mathit{pf}[p, l, b, 1, s] \tag{3}$$

$$\mathit{pl}[p, b] = \mathit{boh}[p, b, t, s] + \mathit{bdi}[p, b, t, s] - \mathit{bdo}[p, b, t, s] \quad \forall t \in [1..T] \tag{4}$$

$$\begin{aligned} \mathit{boh}[p, b, t, s] - \mathit{bdo}[p, b, t, s] = \; & (\mathit{boh}[p, b, t-1, s] - \mathit{bdo}[p, b, t-1, s]) + \\ & \mathit{sdso}[p, b, \mathit{supplier}[p, b], \mathit{max}(t - \mathit{st}[p, \mathit{sd}, b], 0), s] + \\ & \textstyle\sum_{l \in \mathit{fm}[p]} \mathit{rdbr}[p, l, b, \mathit{max}(t - \mathit{rt}[p, l], 0), s] - \\ & \textstyle\sum_{l \in \mathit{fm}[p]} \mathit{pf}[p, l, b, t, s] \\ & \forall t \in [2..T] \end{aligned} \tag{5}$$

Constraint 3 computes the base inventory position at time $t = 1$, when no parts can be received from a supplier but parts can be delivered to repair a downed plane. Constraint 4 implements the standard definition of inventory conservation. While we focus strictly on the $(x, x-1)$ re-order policy, other inventory policies can be implemented via different constraint sets. Constraint 5 computes updates to the base inventory position based on received quantities from supply depots and local repairs, and from additional part failures at a base. For notational convenience, we assume $\mathit{rdbr}[p, l, b, 0, s] = \mathit{sdso}[p, b, \mathit{sd}, 0, s] = 0$.

Supply depots behave similarly to bases in terms of inventory behavior, with additional complexity due to the arrival of inbound shipments from OEMs and repair depots. The constraint sets defining supply depot behaviors are as follows, defined for all $s \in \mathcal{S}$ and $(p, sd) : sd \in \mathcal{SD}$:

$$sdoh[p, sd, 1, s] = pl[p, sd] - \sum_{b \in \mathcal{B}:(p,b) \in \mathcal{PV} \wedge supplier[p,b]=sd} sdso[p, sd, b, 1, s] \tag{6}$$

$$\begin{aligned} sdoh[p, sd, t, s] = \quad & sdoh[p, sd, t-1, s] + \\ & oemso[p, supplier[p, sd], sd, max(t - st[p, supplier[p, sd], sd], 0), s] + \\ & \textstyle\sum_{rd \in \mathcal{RD}:repairdest[p,rd]=sd} rdso[p, rd, sd, max(t - st[p, rd, sd], 0), s] - \\ & \textstyle\sum_{b \in \mathcal{B}:(p,b) \in \mathcal{PV} \wedge supplier[p,b]=sd} sdso[p, sd, b, t, s] \\ & \forall t \in [2..T] \end{aligned} \tag{7}$$

where we assume $rdso[p, rd, sd, 0, s] = 0$. Constraint 6 computes the initial on-hand inventory at a depot; no parts can be received on the first day, but they can be shipped out. Constraint 7 computes updates to the inventory over time, based on inbound shipments from repair depots and OEMs and outbound shipments to bases. Strict inventory conservation is not enforced, as supply depots serve to handle overflow in the system.

Base repair functionality in our SESP formulation is modeled via the following constraint sets, defined for all $s \in \mathcal{S}$ and $(p, l, b) : repairer[p, l, b] = b$:

$$rdwr[p, l, b, 1, s] = pf[p, l, b, 1, s] \tag{8}$$

$$rdwr[p, l, b, t, s] = rdwr[p, l, b, t-1, s] + pf[p, l, b, t, s] - rdbr[p, l, b, t, s] \quad \forall t \in [1..T] \tag{9}$$

For notational convenience, we assume $rdwr[p, l, b, 0, s] = 0$. Constraint 8 models the fact that parts failing at a base immediately enter the repair queue at that base. Constraint 9 computes the number of parts awaiting repair based on local failures and initiated part repairs. It is then necessary to enforce the resource constraints, ensuring that no more than $rl[r, b]$ resources of type r at base b are simultaneously in use, for all $t \in [2..T]$, $s \in \mathcal{S}$, and $(r, b) \in \mathcal{RV} : b \in \mathcal{B}$:

$$\Big(\sum_{(p,l,b):repairer[p,l,b]=b} \Big(\sum_{tt \in [max(t-rt[p,l],2)..t]} rdbr[p, l, b, t, s] * rq[p, l, r] \Big) \Big) \leq rl[r, b] \tag{10}$$

Note that no repairs can begin on day $t = 1$, to prevent optimistic solutions associated with premature processing of part failures that occur late in the day.

Repair depots are modeled similarly to the repair queues at bases, but additionally account for delays in shipping carcasses from bases and shipment of repaired parts to supply depots. The following constraint implements the boundary conditions for repair depots, instantiated for all $s \in \mathcal{S}$ and (p, l, rd) such that $l \in fm[p]$ and $\exists b \in \mathcal{B} : repairer[p, l, b] = rd$:

$$rdwr[p, l, rd, 1, s] = rdbr[p, l, rd, 1, s] = 0 \tag{11}$$

Because ship times are ≥ 1, repair depots cannot initiate repairs at $t = 1$. The count of parts awaiting repair is computed by the following constraint, instantiated for all $s \in \mathcal{S}$ and $(p, l, rd$ such that $\exists b \in \mathcal{B} : repairer[p, l, b] = rd$:

$$\begin{aligned} rdwr[p, l, rd, t, s] = \quad & rdwr[p, l, rd, t-1, s] + \\ & \textstyle\sum_{b \in \mathcal{B}:p \in p[b] \wedge repairer[p,l,b]=rd} pf[p, l, b, max(t - st[p, b, rd], 0), s] - \\ & rdbr[p, l, rd, t, s] \\ & \forall t \in [1..T] \end{aligned} \tag{12}$$

To enforce resource bounds, an analog to Constraint 10 is defined for repair depots. Finally, the following constraint implements the transfer of repaired parts to the appropriate supply depot, instantiated for all $s \in \mathcal{S}$ and (p, rd) such that $\exists b \in \mathcal{B} \wedge l \in \mathcal{L} : \mathit{repairer}[p, l, b] = \mathit{rd}$:

$$\mathit{rdso}[p, \mathit{rd}, \mathit{repairdest}[p, \mathit{rd}], t, s] = \sum_{l \in \mathit{fm}[p]} \mathit{rdbr}[p, l, \mathit{rd}, \mathit{max}(t - \mathit{rt}[p], 0), s] \quad \forall t \in [1..T] \quad (13)$$

In the simplified SESP formulation, the resource constraints allow for an optimistic (i.e., able to initiate repair of those parts it knows will be needed based on failure stream data) repair scheduling policy; intricate constraints are required to implement non-clairvoyant scheduling policies such as FIFO.

The per-base availability performance criteria are computed via:

$$\sum_{(p,b) \in \mathcal{PV}, t \in [1..T]} \mathit{bdo}[p, b, t, s] \leq (1 - \nu[b])\mathit{ad}[b] \quad \forall b \in \mathcal{B}, s \in \mathcal{S} \quad (14)$$

Intuitively, each day there is a part due out from a base to repair a plane, an air day is lost for flying missions. Fundamentally, Constraint 14 is what makes the SESP a robust, as opposed to a stochastic, MIP: the target performance criterion must be achieved in *all* scenarios.

The SESP optimization objective is to minimize the procurement costs associated with the first-stage decision variables, given as:

$$\sum_{(p,a) \in \mathcal{PV}} c[p] \cdot \mathit{pl}[p, a] + \sum_{(r,a) \in \mathcal{RV}} c[r] \cdot \mathit{rl}[r, a] \quad (15)$$

We have considered various second-stage costs, including volume-based warehousing costs, in preliminary versions of this formulation. However, these costs are typically dwarfed by the initial procurement costs, such that the benefit of incorporating second-stage costs is outweighed by the increase in computational difficulty. Similarly, no recourse actions are available to facilitate recovery from poor initial decisions, although this is clearly supported by the PH framework. While not known to us at the time of formulation, our SESP robust model is closely related to Schruben's [19] event graph methodology for constructing MIP models of discrete event systems. Finally, we note that we have omitted discussion of numerous subtle issues that significantly inflate the complexity of the straightforward SESP formulation, including constraints required to prevent clairvoyance induced by the up-front availability of all part failures that will occur during a scenario $s \in \mathcal{S}$ and optimizations to eliminate redundant constraints and variables that are too difficult to be identified during MIP presolve.

As discussed in Section 3.2.5, our robust formulation of the SESP extensive form is extremely difficult to solve via CPLEX 10.1; LP relaxations take hours to days of run-time, and feasible MIP solutions are rarely generated after a week or more of run-time. This difficulty is due to both the size of the formulation (e.g., see Table 3 below) and constraints such as those required to prevent clairvoyant behavior, as discussed in [6]. The solution of individual SESP scenarios also poses a computational challenge, with run-times ranging in the tens of minutes for the moderate-sized test instances introduced below in Section 3.2.3. When used in a multi-iteration, multi-scenario PH context, the aggregate run-times render the approach infeasible. Consequently, we instead deploy powerful, yet simple problem-specific heuristics for solving individual SESP scenarios.

Recall that our SESP formulation is based on a per-day coarse-graining of the SEM simulator time resolution, with additional simplifications relating to the modeling of aircraft maintenance procedures and other site-level operational details. Leveraging this situation, we were able to develop a very high-speed approximate variant of the SEM simulator, written in C++.

Given part failure data for a scenario $s \in \mathcal{S}$, the resulting program can evaluate the performance of a particular choice of first-stage part and resource level decision variables in milliseconds of run-time, where performance is quantified in terms of the operational availability obtained at each base. Such low solution evaluation speeds enable the use of greedy local search procedures to determine near-optimal settings for the first stage decision variables $pl[p, a]$, $(p, a) \in \mathcal{PV}$, and $rl[r, a]$, $(r, a) \in \mathcal{RV}$.

For each scenario $s \in \mathcal{S}$, a static analysis of the part failure data $pf[p, l, b, t, s]$ straightforwardly yields scenario-specific upper bounds for the $pl[p, a]$. Using these maximal $pl[p, a]$, we then run the coarse-grained simulation with $rl[p, a] = \infty$, and determine the maximal $rl[r, a]$ in use at any time step. The combined maximal $pl[p, a]$ and $rl[r, a]$ across all $s \in \mathcal{S}$ serve as the initial starting point for greedy search for a low-cost solution to each specific scenario s. At each step in the greedy search procedure, we compute for each $(p, a) \in \mathcal{PV}$ and $(r, a) \in \mathcal{RV}$ the *total* decrease in availability, summed across all bases, incurred by a unit decrease in the corresponding part/resource level. For those (p, a) and (r, a) in which base availability targets are still achieved, the pair with the smallest fraction γ equal to the total decrease in availability divided by the change in solution cost is determined, with the working solution part or resource level modified accordingly. The procedure continues until there exist no (p, a) and (r, a) for which $pl[p, a]$ and $rl[r, a]$ can be reduced without creating an infeasible solution. To simplify the exposition, we do not discuss numerous details of the greedy procedure relating to the PH cost objective and variable unit decrease factors, which impact overall performance to variable degrees.

Using MIP solutions of individual SESP scenarios obtained using CPLEX 10.1 (with no run-time bound), we have benchmarked the performance of the solutions resulting from our greedy heuristic. In all cases, the greedy heuristic obtained solutions within 5% of the optimal value, and in most cases deviated by no more than 2%. For the largest test instances we consider in our analysis below, the heuristic run-time per scenario did not exceed two minutes.

3.2.3 The Test Problems

We generate synthetic instances of the SESP, partitioned into two groups. All instances are based on a simple echelon network structure consisting of a single repair depot, supply depot, and OEM, in addition to a set $\mathcal{B}$ of bases; however, our general methodology can handle arbitrary echelon structures. Five aircraft are assigned to each of the bases $b \in \mathcal{B}$. Each base flies two aircraft for 4 hours apiece every day, assuming operational aircraft are available. Each aircraft consists of 50 modeled parts with various failure time distributions, including the exponential and wearout distributions commonly reported in the reliability literature. OEM lead times for building replacement parts range from 30 to 120 days. Part repair times range from 5 days (for base-repairable parts) to 120 days (for depot-repairable parts). Part procurement costs range from as little as \$100 to over \$500,000, respectively representing components such as valves and engines. Inter-site transportation times vary from immediately (e.g., from a plane to the containing base) to nearly a week (e.g., when shipping carcasses from a base to a repair depot). The simulation time horizon is over a single year, and the operational availability targets for all bases at all times is 95%. Each test problem contains failure data for a set $\mathcal{S}$ of scenarios, generated via the SEM discrete event simulator (as discussed in Section 3.2.1). This basic operational environment, albeit simple, is inspired by a proprietary real-world data set analyzed by the authors.

In the first test problem group, the decision variables consist of the initial procurement level for each part at each base and supply depot in the system. Repair processes do not require personnel or support equipment, and are completed after a fixed duration. In other words, both the bases and repair depots are non-capacitated, i.e., there is no limit to the number of concurrent repairs and parts enter repair immediately upon receipt at a site. This first problem group was

devised to mirror the assumptions underlying traditional approaches to spare-parts procurement planning, including METRIC [20] and VARI-METRIC [22]. In the second test problem group, each base- and depot-repairable part requires one or more resources to accomplish the associated repair task. Support equipment costs range from $5,000 to $20,000, while annualized personnel pay rates range from under $20,000 to over $60,000. Repairs require the associated resources for the duration of the repair task, and are released upon completion of the repair task; pre-emption is disallowed. The decision variables in this problem group include both the initial procurement levels for spare parts and the number of each resource type at each base and repair depot in the system. This second test problem group was devised to represent the much more difficult "inventory plus repair" sustainment problem [1], which receives comparatively little attention in the logistics and sustainability literature.

For both test problem groups, we consider instances with both $|\mathcal{S}| = 10$ and $|\mathcal{S}| = 30$ scenarios, in addition to $|\mathcal{B}| = 2$, $|\mathcal{B}| = 5$, and $|\mathcal{B}| = 10$ bases; this yields a total of 12 test problems. We note that solutions obtained with $|\mathcal{S}| > 30$ scenarios are not significantly different than those for $|\mathcal{S}| = 30$ scenarios, i.e., $|\mathcal{S}| = 30$ is empirically sufficient – due to the long time horizon and aggressive operational pace – to achieve target performance goals on unobserved scenarios. The parameters for the first problem group are not overly realistic (being modified to disguise the original source data set), principally due to the repair of otherwise inexpensive parts and only moderate correlation between repair times and procurement cost. However, neither factor plays a critical role due to the lack of repair queue modeling in these instances. In contrast, the instances in the second problem group necessarily correct this deficiency, and are consequently much more realistic in terms of their overall behavior. However, they are less representative of the original source data set, which accounts for the differences in cost observed for the two problem formulations (as reported in Sections 3.2.5 and 3.2.6). The number of decision variables for the first problem group ranges between 144 and 528 (due to variance in $|\mathcal{B}|$), and between 157 and 566 for the second problem group.

All test instances are freely available for general use, and can be obtained by contacting the authors. The size of the instances was selected to allow for investigations over a wide range of PH algorithmic settings, many requiring lengthy run-times. Significantly larger, real-world test problems have also been considered. In particular, specific PH variants have been successfully executed on various instances representing the enterprise-level deployment structure of the sustainability support system for the Lockheed Martin Joint Strike Fighter or JSF [9], and limited-scale forms of the US Army's Future Combat System [4]. Although the details are proprietary, we note that the full JSF deployment contains over 3,000 aircraft (each containing thousands of modeled parts) assigned to over 50 bases worldwide, tens of supply and repair depots, tens of OEMs, all arranged in a complex multi-echelon network structure.

3.2.4 Experimental Methodology

As discussed previously, we have developed a MIP model of the SESP robust optimization problem. Ultimately, we moved to deployment of domain-specific heuristics to solve individual scenario sub-problems due to the difficulty of the MIP formulation, in terms of both run-time and memory requirements, even using high-speed, multi-processor, 64-bit workstations running CPLEX 10.1 [8]. However, the MIP formulation is not without use in our analysis, as it can be used to solve relaxed versions of the extensive form of each test instance. In particular, we use the LP relaxation of the SESP extensive form MIP to obtain bounds on the performance of our PH configurations; we denote the corresponding relaxed solutions by x^*. Additionally, we report results relative to the corresponding LP-rounded extensive form solution, in which each decision variable $x(i)$ is rounded to $\lceil x(i) \rceil$ in the case of fractional $x(i)$. Due to limitations in the accuracy of the SESP extensive form MIP (for reasons described in Section 3.2.1), such rounded solutions – which we denote $\lceil x^* \rceil$ – universally fail to achieve the target oper-

$\|\mathcal{B}\|$	$\|\mathcal{S}\|$	Num. Variables	Num. Constraints	Num. Nonzeros
2	10	1,747,974	1,278,454	6,938,447
	30	5,257,738	3,844,156	20,863,637
5	10	3,695,427	2,693,996	16,080,929
	30	11,089,159	8,083,285	48,244,343
10	10	7,265,206	5,286,639	32,822,147
	30	21,777,866	15,848,928	98,401,035

Table 3: Problem size statistics for the SESP extensive form LP relaxation, after invocation of ILOG CPLEX 10.01 presolve.

ational availabilities when assessed in the context of the SEM simulator. In many cases, the resulting availabilities fall far short, e.g., tens of percent, of the required performance targets. Consequently, gaps in the cost of the PH and $\lceil x^* \rceil$ solutions are not necessarily indicative of the inability of our PH algorithm, or the greedy heuristics for solving scenario sub-problems, to locate very near-optimal solutions; by design these solutions achieve the requisite performance targets in the context of SEM, unlike the bounds to which they are compared.

In Sections 3.2.5 and 3.2.6, we describe the results of parameter sensitivity experiments on our PH variants for spares-only and spares-plus-resources test problems. For each problem group and specific value of $|\mathcal{B}|$ and $|\mathcal{S}|$, we execute PH for each combination of fix lag:

$$\mu \in \{0, 1, 2, 5\}$$

and ρ selection strategy in:

$$\{\text{CP}(0.5), \text{CP}(1.0), \text{SEP}, \text{FX}(20K), \text{FX}(100K), \text{FX}(500K)\}$$

For each individual PH run, we set the termination criteria parameters λ_q and λ_t equal to 1% and 0.0001, respectively. The objective of these experiments is to quantify the effectiveness of the various PH algorithmic techniques introduced in Section 2, and to assess the sensitivity of the proposed PH algorithm to specific parameter settings. For each run, we report the lowest-cost solution x^{max} generated during *any* PH iteration k, where s^{max} is formed by taking the element-wise maximum across all scenarios $s \in \mathcal{S}$; s^{max} is guaranteed to be feasible in all scenarios due to the one-sided resource constraints defining the problem class. All runs are executed on a 64-bit AMD Opteron 2.2GHz workstation, with 64GB of RAM (relevant only for LP solves) running Linux 2.6; individual scenario sub-problems require at most 10 Mb of RAM to solve via the greedy heuristic.

3.2.5 Spares-Only Performance Results

We first consider experimental results for the spares-only test instances. In Table 3 we provide statistics for our test instances regarding the variable, constraint, and non-zero constraint coefficient counts for the SESP extensive form LP relaxation, following invocation of ILOG CPLEX 10.1's presolve phase. The magnitude of the LPs is remarkable, especially considering that industrial-scale instances are orders-of-magnitude larger. The LP and LP-rounded solution quality and CPLEX 10.1 solver statistics for each test instance are shown in Table 4. For fixed $|\mathcal{B}|$, we observe no more than 8% growth in solution cost (which occurs when $|\mathcal{B}| = 5$) when moving from $|\mathcal{S}| = 10$ to $|\mathcal{S}| = 30$ scenarios. The increase in solution cost is due to the increased diversity of part failure sequences. Empirically, the growth in cost for these test instances halts near $|\mathcal{S}| = 30$. This is consistent with the fact that PH solutions for a specific set of $|\mathcal{S}| = 30$ scenarios achieve the target performance objectives under disparate sets of scenarios; in other words, performance generalizes to new scenarios. Overall, the solution times

$\|\mathcal{B}\|$	$\|\mathcal{S}\|$	$c \cdot x^*$	$c \cdot \lceil (x^*) \rceil$	Sol. Time	Memory
2	10	54,432,705.04	55,107,950	10.7m	1.4GB
	30	56,586,131.16	57,445,700	48.96m	3.2GB
5	10	117,512,266.69	119,044,750	50.36m	3.3GB
	30	126,277,945.18	126,639,800	328.78m	5.9GB
10	10	326,354,459.20	326,902,300	357.86m	7.0GB
	30	338,255,374.29	340,216,550	6226.48m	$\approx$ 19GB

Table 4: Solution and solver statistics for the SESP extensive form LP relaxation using ILOG CPLEX 10.1.

		Fix Lag			
$\|\mathcal{B}\|$	$\|\mathcal{S}\|$	$\mu = 0$	$\mu = 1$	$\mu = 2$	$\mu = 5$
2	10	CP(0.5)	CP(0.5)	CP(0.5)	CP(1.0)
2	30	CP(0.5)	SEP	CP(1.0)	CP(1.0)
5	10	CP(0.5)	CP(1.0)	SEP	CP(1.0)
5	30	CP(1.0)	SEP	SEP	N/A
10	10	SEP	SEP	SEP	N/A
10	30	CP(0.5)	SEP	CP(0.5)	N/A

Table 5: The ρ selection strategy obtaining the lowest-cost solution for different variable fix lag values, independent of run time.

are moderate for most instances, ranging from a few minutes to roughly six hours. However, run-time peaks for the largest test instance ($|\mathcal{B}| = 10$, $|\mathcal{S}| = 30$) at over 4 days. Such a large run-time is not practical, especially given the comparative size of real-world problems (e.g., JSF) and the difficulty of LP solver parallelization. Of equal concern is the required memory; once $|\mathcal{B}| = 5$, the memory requirements exceed the limits of commonly available hardware. Finally, we reiterate that the solution statistics relate to the cost of solving *only* the LP relaxation; MIP solutions are currently impractical for the $|\mathcal{B}| = 5$ and $|\mathcal{B}| = 10$ test instances.

Next, we compare the performance of the fixed-ρ strategies (FX$(20K)$, FX$(100K)$, and FX$(500K)$) with the variable-ρ strategies (CP(0.5), CP(1.0), and SEP). In Table 5, we record the ρ selection strategy achieving the lowest-cost solution over the range of variable fix lags μ, *independent* of run-time. Table entries with "N/A" indicate the run-times for that particular PH configuration were excessive (ranging past several days), and were not allowed to converge. The results conclusively demonstrate that the variable-ρ strategies dominate the fixed-ρ strategies in terms of solution quality, supporting the hypothesis advanced in Section 2.1; the relative run-times of the configurations is considered below. Although not reported in Table 5, the three variable ρ strategies generally dominate the best fixed ρ strategy, with some exceptions. To assess the quantitative differences in performance, we report in Table 6 the percentage improvement in solution quality of the best variable ρ strategy over the best fixed ρ strategy. While small in absolute terms, this is an artifact of the enormous baseline expense of the logistics chains in the SESP. Fractions of percent translate into millions of dollars of savings, which is very significant in practice.

In limited trials, we experimented with additional values of fixed ρ. Due to the variance in part costs, ranging over several orders of magnitude, smaller fixed values of ρ yielded excessive run-times, while values of ρ greater than 500K yielded monotonically decreasing solution quality. Overall, no single variable-ρ selection strategy dominates in terms of performance. Although there are some apparent patterns in the data, e.g., SEP dominating for large $|\mathcal{B}|$ and $|\mathcal{S}|$,

		Fix Lag			
$\|\mathcal{B}\|$	$\|\mathcal{S}\|$	$\mu = 0$	$\mu = 1$	$\mu = 2$	$\mu = 5$
2	10	0.86	0.64	0.76	1.39
2	30	1.10	0.57	1.16	0.25
5	10	0.77	0.59	0.41	0.21
5	30	1.14	0.59	0.71	N/A
10	10	0.14	0.66	0.31	N/A
10	30	0.61	0.90	0.80	N/A

Table 6: The percentage improvement in solution quality obtained by the best variable ρ strategy over the best fixed ρ strategy, independent of run-time.

	Fix Lag		
ρ Selection Strategy	$\mu = 0$	$\mu = 1$	$\mu = 2$
SEP	0.1790%	0%	0.0034%
CP(0.5)	0.4440%	0.1188%	0.3794%
CP(1.0)	0.4490%	0.3056%	0.0070%
FX$(100K)$	0.3149%	0.7612%	0.5402%

Table 7: Quality of solutions obtained by variable-ρ and FX$(100K)$ strategies on the $|\mathcal{B}| = 10$, $|\mathcal{S}| = 10$ spares-only test problem; values represent the percentage above the cost of the best known solution.

the number of samples is too limited to make any general inference. Further, performance is appears to be dependent on μ. However, the results do clearly illustrate the power of variable-ρ strategies for the SESP relative to the standard fixed-ρ strategies.

To expand on the results presented in Tables 5 and 6, we consider PH performance on the $|\mathcal{B}| = 10$, $|\mathcal{S}| = 10$ spares-only test instance under the ρ selection strategies FX$(20K)$, FX$(100K)$, and FX$(500K)$. In Figure 1, we show plots of ρ versus the following PH performance measures: solution quality, number of iterations, and run-time. As expected, with the noted exception at $\mu = 0$ and $\rho = 100K$, higher-quality solutions are obtained as the fix lag μ (subject to a constant ρ strategy) is increased (Figure 1a). However, the highest-quality solutions are not obtained with smaller ρ values. While $\rho = 500K$ yields lower-quality solutions, there is no clear preference between $\rho = 20K$ and $\rho = 100K$. This is despite the significant increase in run-time (Figure 1c). Overall, the results suggest that $\rho = 100,000$ with either $\mu = 0$ or $\mu = 1$ yields the best solution quality vs. run-time performance tradeoff; detailed analyses of results on the other spares-only test instances supports this general conclusion. Finally, we observe that the total number of PH iterations required for convergence on these large-scale test instances – even using the various acceleration techniques we propose – is quite large, ranging from approximately 50 to over 2000 (Figure 1b).

We now analyze the performance of the variable-ρ strategies in more detail, comparing solution quality and run-time relative to both one another and to the FX$(100K)$ baseline strategy; the latter yielded the best overall performance for a fixed-ρ strategy. Solution quality results for the $|\mathcal{B}| = 10$, $|\mathcal{S}| = 10$ test instance, expressed as a percentage above the cost of the best known solution, are shown in Table 7. The results illustrate that all of the variable-ρ strategies yield comparable solutions in terms of cost, deviating by no more than 0.5% from the best known solutions. Although small in relative terms, the corresponding absolute cost deviations range between \$2 and \$3 million (from the baseline of approximately \$343 million). Consequently,

	Fix Lag		
Variable-ρ Selection Strategy	$\mu = 0$	$\mu = 1$	$\mu = 2$
SEP	375%	456%	312%
CP(0.5)	198%	342%	152%
CP(1.0)	52%	178%	122%

Table 8: Run-time of variable-ρ PH strategies on the $|\mathcal{B}| = 10$, $|\mathcal{S}| = 10$ spares-only test instance, relative to the baseline FX$(100K)$ strategy.

there is practical motivation for identifying a clear winner among the three strategies. Unfortunately, no clear pattern emerges in the analysis of the data in either Table 7 or for the other test instances. In general, the results for the SEP and CP(0.5) strategies are comparable, and together slightly outperform CP(1.0). However, any given variable-ρ strategy may yield the best overall performance on an individual instance. Finally, comparing the variable-ρ strategies with the FX$(100K)$ strategy, we find that although FX$(100K)$ outperforms some of the variable-ρ strategies when $\mu = 0$, it is dominated by all of the variable-ρ strategies for $\mu \geq 1$; further, the $\mu = 0$ results are not replicated on any of the other test instances.

Despite superior performance, the solution quality of the variable-ρ strategies does come with an increase in run-time cost. In Table 8, we show the percentage increase in overall run-time for the variable-ρ strategies relative to the FX$(100K)$ strategy on the $|\mathcal{B}| = 10$, $|\mathcal{S}| = 10$ test instance. The results indicate that variable-ρ strategies can require nearly 5 times the run-time of the fixed-ρ strategies. The SEP strategy typically requires more computation than the CP(0.5) and CP(1.0) strategies, although we note that the latter is parameterless. The CP(1.0) strategy yields high-quality solutions with only a moderate increase in run-time relative to fixed-ρ strategies. The SEP and CP(0.5) strategies yield better solutions than the FX$(100K)$ strategy, but at the cost of even greater run-times. However, we observe that a factor of five or less in run-time is often not considered significant in practical applications, especially for planning-level problems such as the SESP.

In summary, our experimental results for various PH configurations on spares-only test instances support two specific conclusions. First, variable-ρ strategies yield higher-quality solutions than fixed-ρ strategies, at the expense of slightly increased run-times. Second, independent of the ρ selection strategy, increases in the fix lag μ improve solution quality, but again at the expense of run-time. However, the relative increases in run-time are modest, such that adoption of the techniques to applications involving long-term planning can be practical. Viewed from another perspective, our results illustrate a classic quality vs. run-time tradeoff for a family of algorithms, the most appropriate of which can be leveraged depending on the context. In particular, we note that planning-level problems are often solved iteratively as both the model and data are refined. Consequently, high-speed strategies such as FX$(100K)$ with low μ can be used in early iterations where model and data uncertainties are high. In practice, specifically on large-scale data sets such as JSF for which the run-times required to solve scenario sub-problems is high, we generally use $\mu \leq 1$ with a CP(1.0) selection strategy to obtain high-fidelity solutions, and $\mu = 0$ with a FX$(100K)$ selection strategy to obtain low-fidelity solutions.

We conclude our analysis of the spares-only test instances by considering the quality of the solutions resulting from PH relative to the absolute baseline costs reported in Table 4, as opposed to the solution costs obtained by other ρ selection strategies. In Table 9, we report the percentage difference in cost between the *best* solution obtained by one of our ρ selection strategies and both the x^* and $\lceil (x^*) \rceil$ solutions. The data indicate that the cost of the best PH solution is no more than 6% greater than both $c \cdot x^*$ and $c \cdot \lceil (x^*) \rceil$. Recall that both the x^* and $\lceil (x^*) \rceil$ solutions fail to achieve the target availabilities when assessed in the context of

$\|\mathcal{B}\|$	$\|\mathcal{S}\|$	% above $c \cdot x^*$	% above $c \cdot \lceil (x^*) \rceil$
2	10	4.15	2.88
	30	5.09	3.52
5	10	6.28	4.92
	30	6.07	5.77
10	10	5.05	4.87
	30	5.48	4.87

Table 9: Deviation in the cost of the best solution obtained by PH under various ρ selection strategies and the cost of both the x^* and $\lceil (x^*) \rceil$ LP solutions.

the SEM simulator, and therefore serve strictly as lower bounds on the optimal solution cost. Given the degree of underperformance observed for these solutions, we conjecture that the best PH solutions are in fact at most a few percent sub-optimal. Overall, these results indicate that despite the lack of provable convergence behavior to a global optimum, PH is locating very high-quality solutions to the SESP. In other words, even with convergence acceleration techniques, PH can serve as a very effective heuristic for solving large-scale robust integer programs.

3.2.6 Spares Plus Resources Performance Results

As discussed in Section 3.2.1, the overwhelming majority of research on SESP optimization is focused on spares-only problems. This is primarily due to the relative difficulty of spares-plus-resources problems, which is reflected in the growth of the run-times required to solve the LP relaxation of our MIP formulation of the SESP. For example, consider the case with $n = 5$ and $k = 10$. As recorded in Table 4, the spares-only test problem with these dimensions required slightly over 50 minutes and 3 GB of RAM to solve the LP relaxation. In the case of the corresponding spares-plus-resources test problem, the run-time increases to over 3 days and 8GB of RAM. The growth is due to a combination of the additional state tracking variables and corresponding constraints required to monitor resource usage. The growth in difficulty with increases in $|\mathcal{B}|$ and $|\mathcal{S}|$ is also remarkable, e.g., CPLEX ran for over 9 days attempting to solve the LP relaxation of the $|\mathcal{B}| = 10$, $|\mathcal{S}| = 10$ spares-plus-resources test problem (requiring roughly 25GB of RAM) before we terminated execution. Consequently, we were only able to obtain x^* and $\lceil x^* \rceil$ solutions to the spares-plus-resources test instances up to dimension $|\mathcal{B}| = 5$ and $|\mathcal{S}| = 10$. On these instances, the cost of the PH solutions ranges from between 8% and 12% greater than that of the x^* and $\lceil x^* \rceil$ LP-based solutions. Although larger than the deviations observed for the spares-only test instances (as reported in Table 9), there is evidence that the increase is primarily due to the weakening of the LP bound when repair-related resources are introduced; in particular, the performance of the $\lceil x^* \rceil$ solutions for spares-plus-resources problems are significantly worse when assessed via the SEM simulator than that observed for the corresponding spares-only instances.

To avoid replication of the presentation in Section 3.2.5, we simply note that the main experimental conclusions observed for our PH variants on spares-only test instances extend to spares-plus-resources test instances. In particular, we find that variable-ρ strategies generally outperform fixed-ρ strategies, albeit at the expense of moderate increases in run-time. However, for any given $|\mathcal{B}|$ and $|\mathcal{S}|$, a specific fixed-ρ strategy may in exceptional cases outperform a specific variable-ρ strategy. Similarly, increases in the fix lag μ yield marginally better solutions, at the expense of significant increases in run-time; for the larger test problems, $\mu = 0$ and $\mu = 1$ were the only computationally tractable PH variants. Finally, the CP(0.5) and CP(1.0) variable-ρ strategies yielded roughly equivalent performance in terms of solution quality relative to SEP,

although the run-times were generally much lower.

In terms of impacting practice – specifically the development of a high-performance algorithm for solving spares-plus-resources instances of the SESP – the relevant question is: What is the relative run-time required to solve spares-only problems versus spares-plus-resources problems. To answer this question, we consider the instance where $|\mathcal{B}| = 10$ and $|\mathcal{S}| = 10$, under a $\text{CP}(0.5)$ ρ selection strategy with $\mu = 0$. In the spares-only case, our PH algorithm required 117 iterations for convergence, with an aggregate run-time of 173.37 minutes. In contrast, the spares-plus-resources instance required only 105 iterations for convergence, at the cost of 314.11 minutes of run-time. The discrepancy in run-time is attributable to the increase in the number of decision variables (i.e., the number of resources of each type allocated to each site), which in turn increases the scenario sub-problem solve times. In general, the aggregate PH run-time for spares-plus-resources problems is within a factor of 4 relative to the run-time required for the corresponding spares-only test problems. Consequently, and in contrast to our experience in solving the LP-relaxation of our MIP formulation to the SESP, the PH heuristic is both effective and scalable, demonstrating the practicality of the proposed approach to solving combined spares and resource SESP instances.

4 Parallelization and Practical Deployment of PH

High-speed throughput is a key issue in the deployment of any practical optimization algorithm. This would appear to be an issue for PH, even given our proposed acceleration techniques. In particular, the run-times associated with the experiments reported in Sections 3.2.5 and 3.2.6 range from several hours to several days, e.g., on the larger $|\mathcal{B}| = 10$, $|\mathcal{S}| = 30$ test instance. Fortunately, as observed by a number of researchers, PH is trivially parallelized by distributing the solution of scenario sub-problems across distinct processors [3, 21]. Due to the empirically low variability of sub-problem solve times, we have observed speed-ups of at worst $n/2$ on our test problems, where $n \leq |\mathcal{S}|$ is the number of CPUs. Consequently, given a modest compute cluster possessing approximately 30 compute nodes, it is possible to reduce the wall clock solution times of PH on these test instances to approximately an hour. Similar deployment platforms are required to achieve tractable optimization via PH on the industrial JSF data sets.

5 Conclusions and Directions for Further Research

We investigated PH as applied to a class of scenario-based resource allocation problem in which decision variables represent resources available at a cost and constraints enforce needs for sufficient combinations of resources. We described extensive computational experiments that demonstrate the efficacy of the Progressive Hedging scenario-based decomposition algorithm and the various enhancements that we have introduced to facilitate its practical, real-world application. We have developed and motivated a problem-independent method for computing good values of the main PH parameter, ρ, that depends on problem-specific data. We additionally describe techniques for accelerating convergence, and detecting cycling behavior and damping it as appropriate. Our experiments indicate that variable-specific ρ selection strategies outperform scalar ρ strategies that are commonly associated with PH implementations reported in the literature.

Some of the enhancements that we introduce exploit the fact that we are solving resource allocation problems with one-sided constraints, specifically bounding the decision variables from below. Our convergence accelerators - which yield lower run-times with limited impact on quality - rely on such one-sided constraints; the same is true of the termination criteria that we develop. This covers a large and important class of resource allocation problems, but

it remains as future research to extend these PH enhancements or develop analogs for more generally constrained resource allocation problems.

The algorithmic techniques we developed were necessitated and driven by the scale of our test problems, illustrating the broader need for more complex and realistic problems to drive the development of practical stochastic and robust programming solvers. For example, without variable fixing PH requires weeks of computer time for convergence on the spares procurement problem. There is a rich and growing literature on algorithms for solving stochastic and robust problems, but research opportunities abound in the area of solving large-scale problems with integer variables. For situations where scenario sub-problems are tractable, but the extensive form cannot be addressed directly, PH provides a mechanism to find good solutions.

Future research is needed for acceleration methods when there are two-sided constraints and for finding good ρ values for variables that do not have cost coefficients. Furthermore, it would be useful to conduct computational experiments concerning the tradeoff between sub-problem solution quality and run-time and overall solution quality. Advanced schemes involving increasing subproblem solution quality over the PH iterations may be particularly fruitful, but the complex interactions with acceleration methods requires experimental investigation.

Acknowledgments

Sandia is a multipurpose laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

References

[1] P. Alfredsson, "Optimization of Multi-Echelon Repairable Item Inventory Systems with Simultaneous Location of Repair Facilities", *European Journal of Operational Research*, Volume 99 (1997), pp. 584–595.

[2] http://www.ampl.com.

[3] N.J. Berland and K.K. Haugen, "Mixing Stochastic Dynamic Programming and Scenario Aggregation", *Annals of Operations Research*, Volume 64 (1996), pp. 1–19.

[4] http://www.army.mil/fcs.

[5] S.G. Garille and S.I. Gass, "Stigler's Diet Problem Revisited," *Operations Research* 49 (2001).

[6] H. Greenberg, "A Fine-Grained Mixed-Integer Programming Model for Logistics Optimization", Sandia National Laboratories, 2007.

[7] A. Gosavi, "Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning", Springer, 2004.

[8] http://www.ilog.com.

[9] http://www.jsf.mil.

[10] P. Kall and S.W. Wallace. "Stochastic Programming", Wiley, Chichester, 1994.

[11] O. Listes and R. Dekker, "A Scenario Aggregation Based Approach for Determining a Robust Airline Fleet Composition", *Transportation Science*, Volume 39 (2005), pp. 367-382.

[12] A. Løkketangen and D. L. Woodruff, "Progressive Hedging and Tabu Search Applied to Mixed Integer (0,1) Multistage Stochastic Programming," *Journal of Heuristics*, Volume 2 (1996), pp. 111–128.

[13] M.H. van der Vlerk, *Stochastic Integer Programming Bibliography*, `http://mally.eco.rug.nl/index.html?spbib.html`, 1996-2003.

[14] J.A. Muckstadt, "Analysis and Algorithms for Service Parts Supply Chains", Springer, 2005.

[15] J.M. Mulvey and H. Vladimirou, "Applying the progressive hedging algorithm to stochastic generalized networks", *Annals of Operations Research*, Volume 31 (1991), pp. 399–424.

[16] G.R. Parija, S. Ahmed, and A.J. King, "On Bridging the Gap Between Stochastic Integer Programming and MIP Solver Technologies," *INFORMS Journal on Computing*, Volume 16 (2004), pp. 73-83.

[17] R.T. Rockafellar and R. J-B. Wets, "Scenarios and policy aggregation in optimization under uncertainty," *Mathematics of Operations Research*, 1991, pp. 119–147.

[18] A. Ruszczyński, "Probabilistic Programming with Discrete Distributions and Precedence Constrained Knapsack Polyhedra", *Mathematical Programming*, Volume 93 (2002), pp. 195–215.

[19] E.L. Savage and L.W. Schruben and E. Yucesan, "On the Generality of Event-Graph Models", *INFORMS Journal on Computing*, Volume 17 (2005), pp. 3–9.

[20] C.C. Sherbrooke, "METRIC: A Multi-Echelon Technique for Recoverable Item Control", *Operations Research*, Volume 16 (1968), pp. 121–141.

[21] A. Silva and D. Abramson, "Computational Experience with the Parallel Progressive Hedging Algorithm for Stochastic Linear Programs", *Proceedings of the 1993 Parallel Computing and Transputers Conference*, pp. 164–174, 1993.

[22] F.M. Slay, "VARI-METRIC: An approach to modeling multi-echelon resupply when the demand process is Poisson with a gamma prior", Report AF501-2, Logistics Management Institute, Washington, D.C., 1984.

[23] F.M. Slay and R.M. King, "Prototype Aircraft Sustainability Model", Logistics Management Institute, Washington, D.C. Report AF601-R2 (1987).

[24] V.D. Smith, D.G. Searles, B.M. Thompson, and R.M. Cranwell, "SEM: Enterprise Modeling of JSF Global Sustainment", *Proceedings of the 37th Conference on Winter Simulation*, 2006, 1324–1331.

[25] R. J-B. Wets, "The Aggregation Principle in Scenario Analysis and Stochastic Optimization", in *Algorithms and Model Formulations in Mathematical Programming*, S.W. Wallace ed., Springer, Berlin, 1989.

[26] D.L. Woodruff and E. Zemel, "Hashing Vectors for Tabu Search", *Annals of Operations Research*, Volume 41 (1993), 123–137.

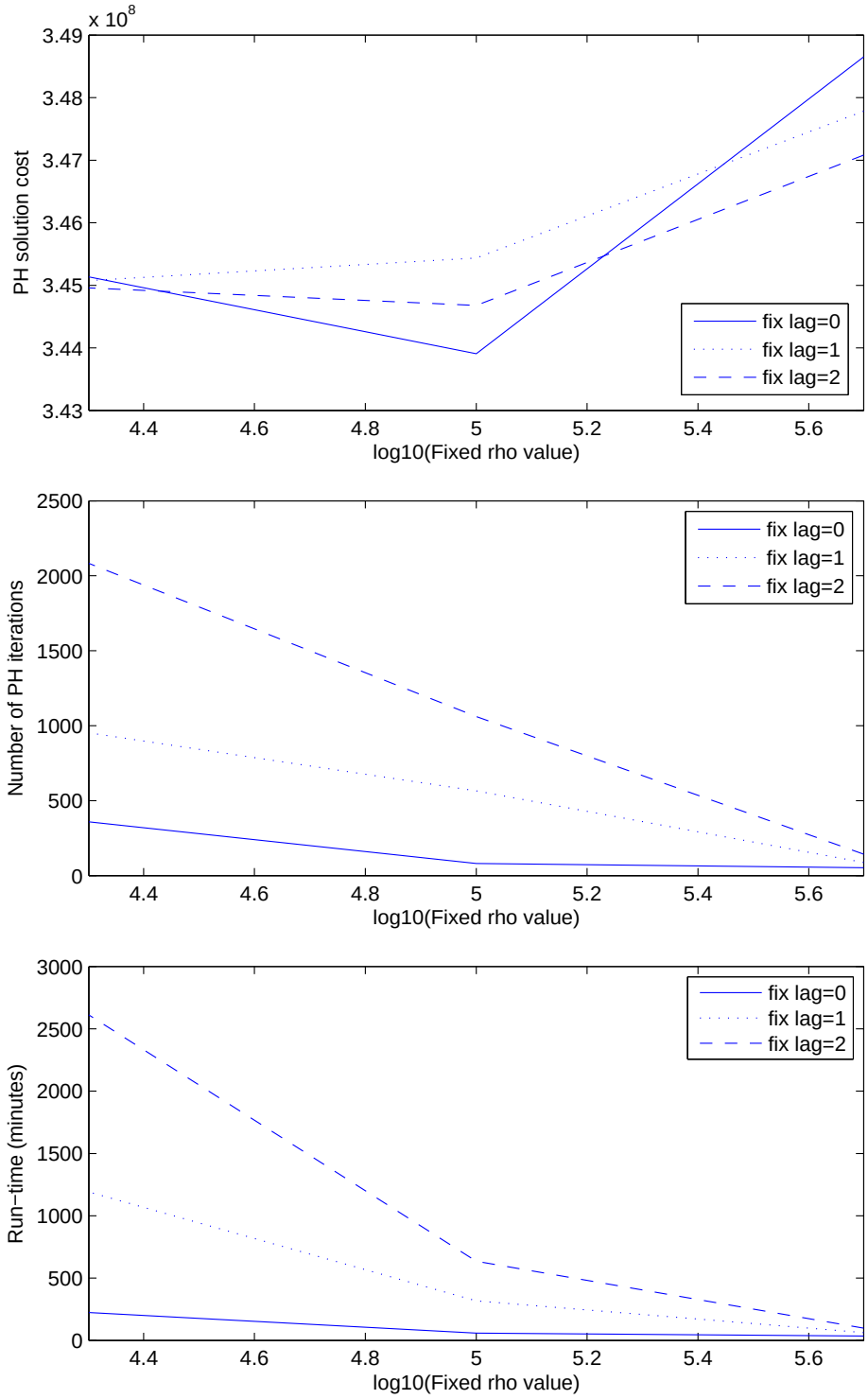

Figure 1: The performance of fixed-ρ PH variants on the $|\mathcal{B}| = 10$, $|\mathcal{S}| = 10$ spares-only test problem. The log-linear plots in the upper, middle, and lower figures display ρ versus (a) solution quality, (b) the number of PH iterations, and (c) run-time, respectively.