

Progressive Hedging Innovations for a Stochastic Spare Parts Support Enterprise Problem

Jean-Paul Watson,¹ David L. Woodruff,² and David R. Strip³

Sandia National Laboratories¹
P.O. Box 5800, MS 1318
Albuquerque, NM 87185-1318 USA
jwatson@sandia.gov

Graduate School of Management²
University of California, Davis
Davis, CA 95616-8609 USA
dlwoodruff@ucdavis.edu

Sandia National Laboratories³
P.O. Box 5800, MS 1316
Albuquerque, NM 87185-1316 USA
drstrip@sandia.gov

Abstract

Progressive hedging (PH) is a scenario-based decomposition technique well-suited to solving stochastic mixed-integer 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 large-scale problems. In particular, decisions must be made regarding the value of the perturbation parameter, ρ , criteria for convergence, and techniques for accelerating convergence. We investigate these issues in the context of a large-scale, real-world stochastic mixed-integer problem for minimizing the procurement costs associated with spare-parts support enterprises. We introduce 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 categories of test problems: those in which only spare-parts procurement levels are considered, and those that additionally consider procurement of repair-related resources. The latter class of problem represents a significant open challenge in the literature, which we show to be efficiently and effectively solved via our PH implementations. Additionally, we demonstrate that variable-specific ρ values are more effective than traditional fixed ρ values, and that the PH algorithm can serve as a very effective heuristic even when the mathematical conditions for convergence are not respected.

1 Introduction

When confronted with a very large, mixed integer stochastic programming problem [10] for which there exists fast, good heuristics for solving individual scenarios, the progressive hedging algorithm proposed by Rockafellar and Wets [17] is appropriate. Progressive hedging (PH) is sometimes referred to as a *horizontal decomposition* method because it decomposes the problem by scenarios rather than by time stages. In this paper we report on innovations that improve the performance of the baseline PH algorithm and on computational experiments with large-scale, real-world problem instances.

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

$$\begin{aligned} & \text{minimize} && c \cdot x_s && (\mathbf{P}_s) \\ & \text{subject to:} && x_s \in \mathcal{Q}_s \end{aligned}$$

where x_s is a decision vector of length n_s , c is a vector of cost coefficients, 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.

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 realized. All real-world decision makers must make a decision even though *a priori* they are not sure which scenario will be realized. The optimization model must therefore possess some mechanisms 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 goal is minimize expected investment cost, which can be written

$$\begin{aligned} & \text{minimize} && \sum_{s \in \mathcal{S}} \Pr(s)(c \cdot x) && (\text{EF}) \\ & \text{subject to:} && x \in \mathcal{Q}_s \end{aligned}$$

where the use of the decision vector x ($x_s = x, \forall s \in \mathcal{S}$) that does not depend on the scenario implicitly implements the *non-anticipativity* constraints that avoid allowing the decisions to depend on the scenario.

For such an optimization problem, the basic PH algorithm can be stated as follows, taking a perturbation factor $\rho > 0$ as the sole input parameter:

1. $k := 0$
2. For all scenario indices $s \in \mathcal{S}$

$$x_s^{(0)} := \underset{x}{\operatorname{argmin}}(c \cdot x) : x \in \mathcal{Q}_s$$

3. $\bar{x}^{(0)} := \sum_{s \in \mathcal{S}} \Pr(s)x_s^{(0)}$
4. $w_s^{(0)} := \rho(x_s^{(0)} - \bar{x}^{(0)})$
5. $k := k + 1$

6. For all scenario indices $s \in \mathcal{S}$

$$\begin{aligned} x_s^k &:= \operatorname{argmin}_x (c \cdot x) \\ &\quad + w_s^{(k-1)} \cdot x + \rho/2 \|x - \bar{x}^{k-1}\|^2 \\ &: x \in \mathcal{Q}_s \end{aligned}$$

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

and

$$\bar{x}^{(k)} := \sum_{s \in \mathcal{S}} \Pr(s) x_s^{(k)}$$

7. If the termination criteria are not met, then go to step 5.

The termination criteria are based mainly on the convergence of the $x_s^{(k)}$ to a common \bar{x} , but we must also deal with the fact that non-convergence is a possibility for non-convex optimization problems, as discussed below.

Integer constraints on elements of the decision vector x 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 problem instances, standard mixed-integer programming (MIP) solvers can be used [16] to directly solve the *extensive form* (EF) of the problem. However, for the problem instances of interest to us, standard MIP solvers can not even reasonably be used to solve individual scenario sub-problems, let alone the extensive form of the problem.

In contrast, PH is a natural 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 the version of the instances with integer restrictions relaxed and then round at the end [11], although for the problem we consider this technique 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 procurement 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 matrices 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 often 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 problem we address in this paper resides in this class, and we propose and demonstrate three methods of accelerating convergence of PH for solving this class of problem.

The next section (2) describes a particular real-world problem from this class that we use as a test case. In Section 3 we describe innovations to PH that allow us to compute a value for the parameter ρ that is based on the input data, speed convergence, detect non-convergence, and terminate PH based on prospects for further improvement. We conclude in Section 5 with a discussion of the impact of our results, in addition to directions for further research.

2 A Spare Parts Support Enterprise Problem

We now detail the specific stochastic mixed-integer problem used in our PH tests. Sections 2.1 and 2.2 respectively introduce the real-world problem upon which our analysis is based and the formulation of and solution techniques for individual scenarios.

2.1 Background

A central problem in aviation fleet management is the minimal-cost sustainability of deployed aircraft [14]. In military contexts, this requirement is typically expressed as the need to ensure a minimal fraction of aircraft in each squadron - for example 95% - are always available for operations. The proportion of available aircraft in a squadron - also known as *operational availability* - is locally (e.g., at a base) driven by two primary factors: availability of resources to perform aircraft maintenance and availability of spare parts to replace failed aircraft components. At the enterprise level, the availability of spare parts to supply demands (e.g., those imposed by bases) is influenced by the capacity of repair depots to fix failed parts, the initial stock and re-order policies of supply depots (which are equivalent in the case of commonly used $(s, s - 1)$ re-order policies), and the lead time required to procure new replacement parts. In practical terms, the sustainability cost is thus dictated by the cost associated with procuring both the initial stocks of spare parts and the quantity of 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 / support problem extends well beyond aircraft, to systems including military ground combat brigades, oil rigs, computing infrastructures, and commercial trucking companies.

A family of analytic optimization models for such sustainability problems - with an emphasis on military aircraft fleets - has been developed over the last 40 years, beginning with METRIC [19] and more recently with ASM [22]. While powerful, especially in terms of their ability to quickly deliver minimal-cost sustainability solutions, these models generally operate under a large set of assumptions, e.g., fixed repair turn-around times and specific parametric part failure distributions. Recently, many military and commercial customers have deemed these restrictions unrealistic and, as a result, have turned to simulation as a method to model their sustainability enterprises. One example of note is the Support Enterprise Model (SEM) [23], 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 various assumptions present in the METRIC-like analytic optimization models and facilitating the deployment of complex business rules, simulation models such as SEM lack an inherent optimization capability. This paper addresses the development of a stochastic programming formulation of the SEM sustainability optimization problem, and introduces a stochastic programming algorithm based on PH to solve the formulation.

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 enterprise; relevant sites include all supply depots (independent of echelon level), OEMs, 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 mainte-

nance, e.g., technicians or engine lifts, are provided through an external analysis tool, and are considered fixed in the current SEM context. Although not required by SEM, we assume for present context 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 the per-item procurement costs associated with the various part/site and resource/site combinations. We denote this general class of optimization problem as Support Enterprise Sustainability Problems, or SESP.

Given a problem instance, we use the SEM simulator to generate part failure data at each squadron in the system for n independent replications or scenarios. These replications are executed in a “flooded” mode, i.e., one in which the supply of parts and available resources is unconstrained. By directly leveraging SEM in this manner, it is possible to sample part failures from non-parametric or not generally accessible (from an analytic standpoint) parametric distributions, e.g., complex wear-out distributions or failure due to 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 could not occur. However, given the typically high availability requirements (on the order of 90% through 95%) for JSF and other military fleets, the degree of conservatism is in practice not significant.

We observe that although the input data for our SESP optimization problem derives from a simulation, we are not using the simulator to compute either the constraints or objective function value associated with a solution. There is a rich literature on this topic (simulation-based optimization) [7]. However, such “black-box” approaches are not feasible for our problem; individual evaluations require minutes to hours of runtime, and the number of decision variables is very large. Although it is not immediately relevant in the context of this paper, simulators such as SEM are in practice used as a final verification step for the solutions we obtain, i.e., the ultimate quality of our resulting solutions is assessed via discrete-event simulation. In summary, high-fidelity simulation provides the input and final validation of our solutions, but is not used in the process of actually generating those solutions.

2.2 Mixed-Integer Formulation and Heuristic Solution Alternatives

For any given SESP scenario, s , it is conceptually straightforward to develop a MIP formulation to express the cost minimization of the support enterprise, subject to the constraint that the average daily operational availability of each squadron over the time horizon of the simulation is greater than or equal to a user-specified threshold. Such a MIP must track state variables such as on-hand and due-out inventory quantities, repair queue and repair in-process status, and the number of aircraft downed due to lack of a spare part. Constraints in the MIP then conserve inventory position across time, enforce limits on the utilization of repair resources, and model inter-site transport delays. In practice, there exist numerous subtle issues that significantly inflate the complexity of the straightforward MIP formulation, such as the need to prevent clair-

voyance induced by the up-front availability of all part failures that will occur during the scenario s .

We have developed such a MIP for the SESP; due to the length of the model, we defer to [6] for a complete description. The model (expressed in the AMPL [2] mathematical programming language) is available from the authors by request. Although it was not known at the time of construction, we observe that our formulation is a large-scale instance of Schruben’s [18] event graph methodology for MIP models of discrete event systems. Because the decision variables in the SESP are common across all scenarios, an extensive form EF of the SESP can be easily constructed by replicating the constraints and scenario-specific (state) variables for each of n scenarios.

Ideally, it would be possible to solve the extensive form of the SESP stochastic program directly with existing commercial MIP solvers. However, this is not currently feasible. Because the aforementioned state variables must be tracked on a daily (or higher-resolution) basis, the size of the resulting MIP - even for relatively small enterprises, let alone the full-scale JSF deployment - is very large. As reported below in Section 4, the memory requirements using commercial MIP solvers frequently exceeds the 4GB RAM limit for 32-bit computing architectures, and reaches nearly 20GB for moderate-scale problems. Additionally, the solve times for even the LP relaxation can extend to several days on powerful 64-bit workstations; MIP solves are therefore not practical in the foreseeable future.

Despite the observed difficulties, failure to solve the extensive form of the SESP does not necessarily preclude usefulness of the MIP formulation. Given the ability to efficiently solve the SESP for one scenario, it is theoretically possible to generate approximate solutions to the extensive form using scenario-based decomposition methods such as PH. However, for the real-world instances considered here, the MIP solves for a scenario are too expensive to be practical. In particular, individual-scenario MIP solves for the moderate-sized problem instances introduced below in Section 4 requires tens of minutes using commercial MIP solvers. Consequently, when used in a multi-iteration, multi-scenario PH context, the resulting solution times for the SESP are highly impractical for full sized, real-world instances.

Based on this observation, and following lengthy attempts to achieve a scalable MIP formulation, we devoted significant effort to develop powerful domain-specific heuristics for solving individual-scenario variants of the SESP. These heuristics, based on randomized descent strategies and coarse-grained simulation (which can compute the implications of specific, arbitrarily complex business rules), can generate solutions to even large-scale SESP instances in at most minutes of run-time, and are used to generate the experimental PH results described in Section 4. The details of these heuristics are not relevant in the context of broader PH performance, the analysis of which is our objective in this paper. Rather, we simply observe - as quantified in Section 4 - that the performance of our heuristics is excellent, achieving near-optimal solutions in significantly lower run-times than obtained using our MIP formulation.

3 PH Algorithmic Innovations

Our experience applying PH to large, real-world problems led us to a number of algorithmic enhancements to the basic algorithm, which can be sub-divided into the following three categories: effective ρ value computation (Section 3.1), convergence accelerators (Section 3.2), and termination criteria (Section 3.3). We additionally describe techniques for detecting cycling behavior in Section 3.4.

3.1 Computing Effective ρ Values

Early - and some recent - experiments concerning PH reported in the literature generally use fairly small values of the perturbation parameter ρ . For example, Mulvey and Vladimirov [15] report that the best values of ρ were much less than 1 and that performance was sensitive to the choice of ρ . However, this need not always be the case. The “best” value of ρ is clearly data dependent. For example, Listes and Dekker [11, p. 374] observe that “there is no conclusive theoretical analysis to support a general selection rule for $[\rho]$ ”. In their experiments, the best results were obtained using ρ values between 50 and 100.

In the context of the SESP, 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 i of the cost vector c may range in magnitude from several hundred dollars to several hundred thousand dollars per unit inventory or resource element. 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 $w_s(i)$ (Step 4 of the pseudocode presented in Section 1) will yield a small fraction of $x(i)$ - whose value represents a quantity, typically less than 100 in the SESP - 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 the SESP.

Based on these observations, we have developed a novel and simple method of determining element-specific ρ_i values based on problem-specific data. As demonstrated in Section 4, the method results in substantially improved PH performance relative to constant ρ values, and partially alleviates the need for problem-dependent parameter tuning. In particular, the method is not based on the SESP, and is therefore applicable in other problem domains.

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 the optimal 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. This 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 precipitate 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} . 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} 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

$$\rho_s := \frac{c}{|x_s - \bar{x}|}$$

However, using a value of ρ that depends on s will violate the convergence assumptions made by Rockafeller and Wets. Furthermore, we want the absolute value of all w elements to approach their ultimate value from below to help mitigate cycling. 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 . We denote this heuristic method for selecting per-element $\rho(i)$ by *sep*.

The primary advantage of the ρ selection heuristic *sep* is its problem-independent nature. 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 the SESP. 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. In Section 4, we report results for 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_k , where *cp* stands for *cost-proportional*. Intuitively, the number of PH iterations required for convergence under this method is proportional to k . Finally, as a control measure, we consider the performance of PH using various fixed, global values of ρ . For a given constant k , we denote the corresponding method by f_k , where the f stands for *fixed*.

3.2 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 optimization problems.

The following three acceleration methods are designed for one-sided diet problems, 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 diet problems, as we will demonstrate, the methods are however quite effective.

A detailed analysis of PH algorithm behavior on the SESP and other problems indicates that individual decision variables $x_s(i)$ frequently converge to specific, fixed value 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 iterations. Such variable “fixing” behaviors lead to an obvious, albeit potentially powerful heuristic: once $x_s^{(k)}(i) = z$ for all $s \in S$ at a particular PH iteration k , fix $x_s^{(l)}(i) = z$ for all subsequent iterations $l > k$. As shown in Section 4, 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 for both individual sub-problems and the final PH solution.

In applying this heuristic to the SESP in particular, we introduce a *lag* parameter $\mu \in \{0, 1, \dots\}$. Consider a given PH iteration k . We then fix $x_s^{(k)}(i)$ for all subsequent iterations $l > k$ once $x_s^{(m)}(i) = z$ for all $s \in S$ and $m \in \{k - \mu|S|, \dots, k\}$, such that $m \geq \mu|S|$. In other words, we fix decision variables once their value has stabilized to a fixed z over the last $\mu|S|$ 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 $|S|$ accounts for the observation that the number of PH iterations required for convergence in general is proportional to the total number of scenarios under consideration.

This idea can be taken further by fixing values for integers 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, widely 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 condition of sufficient convergence has been achieved after some PH iteration k (the specific criteria are described next in Section 3.3), we first set the lag $\mu = 0$, independent of its current value. Then, every κ subsequent iterations we identify the free decision variable x_i for which the per-element cost $c(i)$ is minimal. We then fix $x_i = \max_{s \in S} x_s(i)$. Given our one-sided diet formulation, feasibility of the scenario sub-problems is necessarily not lost via such a maximum-value scheme. Clearly, this scheme is guaranteed for force PH convergence. We have investigated performance using various κ , including $\kappa = |S|$. However, we found minimal performance gains using such large κ , at the expense of significantly increased run-times; consequently, we fix $\kappa = 2$ in the experiments reported below in Section 4.

3.3 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 average per-scenario deviation from the “average” solution $td = (\sum_{i,s} x_s(i) - \bar{x}(i))/|\mathcal{S}|$, where $\bar{x}(i)$ 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, in the SESP there often exist "holdout" scenarios that require more high-cost parts - for example, engines - than other scenarios. Further, these additional high-cost parts are required to achieve feasibility in the holdout scenarios. 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 part / 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., $x^{max}(i) = \max_{s \in \mathcal{S}}(x_s^{(k)}(i))$. Clearly, x^{max} is a feasible (albeit likely suboptimal) solution to all scenarios $s \in \mathcal{S}$. Finally, let $qd = (c \cdot x^{max} / c \cdot \bar{x}) * 100$. 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 implementation for solving the SESP, we invoke variable slamming once *both* $td \leq \lambda_t$ and $qd \leq \lambda_q$ after an iteration k .

3.4 Detecting Cyclic Behavior

Finally, we note that for all types of non-convex optimization problems, there is a risk of non-convergence of the PH algorithm. This is due in part to the use of various convergence acceleration techniques, including those described previously; furthermore, the basic PH algorithm introduced by Rockafellar and Wets only guarantees eventual convergence to a local optimum in the non-convex case. In the experiments discussed in Section 4, non-convergence occurs in roughly one tenth of all algorithmic trials. In particular, we observe non-convergence in the form of cyclic behavior across different PH iterations, e.g., repeated identical weight and decision vectors $w_s(i)$ and $x_s(i)$ for specific elements i . To detect cycles, we chose to focus on repeated occurrences of $w_s(i)$ vectors, implemented using a hashing scheme [24] 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 diet problems. In practice, few variables are fixed in such a fashion, yielding minimal impact on final solution quality while assuring termination.

4 Experimental Analysis of PH Performance

We now perform a comprehensive empirical performance analysis of the various PH algorithmic techniques described in Section 3, using the SESP as a test-bed. Section 4.1 describes the problem instances, which are based on a proprietary real-world data set. The experimental methodology is presented in Section 4.2. Performance results on spares-only and spares-plus-resources test cases are respectively detailed in Sections 4.3 and Section 4.4. We conclude in Section 4.5 with a brief discussion concerning parallelization of the PH algorithm to yield significant run-time reductions in a deployment environment.

4.1 The Test Problems

We quantify the performance of our PH variants using two categories of test problems. The instances in both categories are based on a simple echelon network structure consisting of a single repair depot, supply depot, and OEM, in addition to n operational bases; however, our general methodology can handle arbitrary echelon structures. Five aircraft, composed in a single squadron, are assigned to each of the n bases. Each squadron flies a single sortie consisting of two aircraft - assuming they are functional - for 4 hours every day. Each aircraft consists of 50 modeled parts, representing a range of failure distributions (e.g., random and wearout) experienced during operational flying time. A failure mode is associated with each distinct part type: consumable, base-repairable, and depot-repairable. Upon failure, a consumable part (e.g., a tire) is immediately disposed of, while base- and depot-repairable parts (e.g., engines) enter the repair queue at the respective locations. Replacement consumable parts are built by the OEM, with lead times ranging 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 around \$100 to over \$500,000, respectively representing components such as tubes and engines.

Parts repaired at a base immediately re-enter inventory at that base; parts repaired at the repair depot enter inventory at the sole supply depot after shipment. Each base requests additional inventory from the supply depot, which in turn can request new OEM builds for consumable parts. A simple $(s, s - 1)$ stocking policy is assumed, and we assume unitary batch size for new builds. 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 optimization objective is to maintain a high operational tempo in each scenario - specifically 95% availability of the aircraft in each squadron - at the minimal cost required to procure the initial inventory of spare parts and the acquisition cost of any resources; the latter includes operational costs incurred during the simulation period. The subsequent costs associated with building replacements for consumable parts and any shop material required to repair other parts are not treated in the course of optimization, as these are considered "sunk" costs, i.e., the actions must be performed irregardless. This basic operational environment, albeit simple, is inspired by a proprietary real-world data set analyzed by the authors for a distinct analysis and customer. Finally, each test problem contains failure data for k realizations or operational scenarios. The data for each realization is generated via the SEM discrete event simulator, as discussed in Section 2.

In the first category of test problem, the decision variables consist of the initial procurement level for each part at each base in the system, in addition to those at the supply depot. Because we assume one-for-one replenishment, these variables are equivalent to the re-order up-to levels s . Repair processes do not require personnel or support equipment, and are completed after a fixed duration. In these problems, 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. The first category of test problem was devised to mirror the assumptions underlying traditional approaches to spare-parts management, including METRIC [19] and VARI-METRIC [21] (non-capacitated depots with a fixed repair duration d_1 are equivalent to capacitated depots with a fixed repair duration $d_2 > d_1$). These assumptions are relaxed in the second category of test problem, in which each base- and depot-repairable part requires one or more resources to accomplish the repair task. Resources are subdivided into two categories: personnel and support equipment. 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. The decision variables in this category of test problem 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. The second category of test problem was devised to represent the much more difficult class of "inventory plus repair" sustainment problem [1], which receives comparatively little attention in the logistics and sustainability literature.

For both the spares-only and spares-plus-resources categories, we consider test problems with both $k = 10$ and $k = 30$ scenarios, in addition to $n = 2$, $n = 5$, and $n = 10$ bases. This yields a total of 12 test problems. We note that solutions obtained with $k > 30$ scenarios are not significantly different than for $k = 30$ scenarios, i.e., $k = 30$ is sufficient for these instances (due to the long time horizon and heavy operational pace) to achieve target performance on unobserved scenarios. The parameters underlying the spares-only test problems are not overly realistic (being heavily modified to disguise the original source data set), principally due to the repair of otherwise inexpensive parts and strictly moderate correlation between repair times and procurement cost. However, neither factor plays a critical role due to the lack of repair queue modeling. In contrast, the spares-plus-resources test problems 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 classes (as reported in Sections 4.3 and 4.4. The number of decision variables for the spares-only problems ranges between 144 and 528, and between 157 and 566 for the spares-plus-resources problems. In this particular formulation, no recourse (due to real-world constraints imposed by the target enterprises) is possible from poor initial decisions, although this is clearly supported by the PH framework.

All of the test problems are freely available for general use, and can be obtained by contacting the authors. The size of the test problems was selected to allow for investigation of a wide range of PH algorithmic settings, many requiring lengthy run-times. Significantly larger, real-world test problems have also been investigated. In particular, specific PH variants have been successfully executed on various test problems representing the enterprise-level deployment structure of the 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 sensitive, 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.

4.2 Methodology

As discussed previously, we have developed MIP models of the SESP 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 10.0 [8]. However, the MIP formulation is not without use in our analysis, as it can be used to solve relaxed forms of the Extensive Form (EF) of each test problem. In particular, we use the LP relaxation of the EF MIP to bound the performance of the PH heuristic below in Sections 4.3 and 4.4; we denote the corresponding relaxed solutions by x^* . Additionally, we report results relative to the corresponding LP-rounded solution, in which each decision variable is set to the

next highest integer value in the case of fractional variables. Due to limitations in the accuracy of the MIP formulation (for reasons described in Section 2), such rounded solutions - which we denote $\lceil x^* \rceil$ - fail to achieve the target operational 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 targets. Consequently, gaps in the cost of the PH and $\lceil x^* \rceil$ solutions are not necessarily indicative of the inability of the PH heuristic - or the greedy heuristic for scenario sub-problems - to locate very-near or even optimal solutions; by definition these solutions achieve the requisite performance targets in the context of SEM.

In Sections 4.3 and 4.4, we describe the results of parameter sensitivity experiments on our PH variants for spares-only and spares-plus-resources test problems. For each problem category and specific value of n and k , we execute PH for each combination of variable fixing lag

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

and $\rho(i)$ selection strategy

$$\kappa = \{cp_{0.5}, cp_{1.0}, sep, f_{20K}, f_{100K}, f_{500K}\}$$

For each individual PH run, we set the termination criteria parameters λ_q and λ_t equal to 0.5% and 0.5, respectively. The objective of these experiments is to quantify the effectiveness of the various PH algorithmic techniques introduced in Section 3, and to assess the sensitivity of the PH algorithm to specific parameter settings. For each run, we report the lowest-cost “maximum” solution x^{max} generated during *any* PH iteration k . Although x^{max} generally equals the x obtained by PH at convergence, differences do occur in a minority of runs. Such discrepancies are possible because the algorithmic techniques that accelerate (and thereby interfere with) the standard PH algorithms may in fact lead to convergence to a solution that is not locally optimal. All runs are executed on a quad-processor 64-bit AMD Opteron 2.2GHz workstation, 64GB of RAM (relevant only for LP solves) running Linux 2.6; individual scenario sub-problems require at most ten megabytes of memory to solve via the greedy heuristic.

4.3 Spares-Only Performance Results

Num. Bases (n)	Num. Scenarios (k)	$c \cdot x^*$	$c \cdot \lceil x^* \rceil$	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 1: Solution and solver statistics for Extensive Form SESP LP relaxation using ILOG CPLEX 10.0.

We first consider experimental results for the spares-only test problems. The LP and LP-rounded solution quality and CPLEX 10.0 solver statistics for each test problem are shown in Table 1. For fixed n , we observe no more than 8% growth in solution cost (which occurs when $n = 5$) when moving from $k = 10$ to $k = 30$ scenarios. The increase in solution cost is due to the increased diversity of part failure sequences.

Empirically, the growth in cost for these test problems halts near $k = 30$; this is consistent with the fact that PH solutions for a specific set of $k = 30$ scenarios achieve target performance objectives under disparate sets of scenarios; in other words, their performance generalizes to new scenarios. Overall, the solution times are moderate for most problems, ranging from a few minutes to roughly six hours. However, run-time peaks for the largest test case ($n = 10, k = 30$) at over 4 days. In practice, 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 $n = 5$, the 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; integer-feasible solutions are currently impractical for the $n = 5$ and $n = 10$ test problems.

		Fix Lag			
Num. Bases (n)	Num. Scenarios (k)	$\mu = 0$	$\mu = 1$	$\mu = 2$	$\mu = 5$
2	10	$cp_{0.5}$	$cp_{0.5}$	$cp_{0.5}$	$cp_{1.0}$
2	30	$cp_{0.5}$	<i>sep</i>	$cp_{1.0}$	$cp_{1.0}$
5	10	$cp_{0.5}$	$cp_{1.0}$	<i>sep</i>	$cp_{1.0}$
5	30	$cp_{1.0}$	<i>sep</i>	<i>sep</i>	N/A
10	10	<i>sep</i>	<i>sep</i>	<i>sep</i>	N/A
10	30	$cp_{0.5}$	<i>sep</i>	$cp_{0.5}$	N/A

Table 2: The ρ selection strategy obtaining the lowest-cost solution for different lag values, independent of run time.

Next, we compare the performance of the fixed- ρ selection strategies (f_{20K} , f_{100K} , and f_{500K}) with the variable- ρ selection strategies ($cp_{0.5}$, $cp_{1.0}$, and *sep*). In Table 2, we record the ρ selection strategy achieving the lowest-cost solution over the range of variable fixing 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 run to completion. The results conclusively demonstrate that the variable- ρ strategies dominate the fixed- ρ strategies in terms of final solution quality, supporting the hypothesis advanced in Section 3.1; the relative run-times of the methods is considered below. In limited trials, we experimented with additional values of fixed ρ . Due to disparities with average part costs, 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 n and k , the number of samples is too limited to make an accurate inference. Further, performance is not clearly 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.

Although the quality of fixed- ρ solutions is worse than that of variable- ρ solutions, we have not considered the relative run-times of the two approaches, nor quantified the relative performance of the various fixed- ρ strategies. To expand on the results presented in Table 2, we consider PH performance on the $n = 10, k = 10$ spares-only test problem under the ρ selection strategies f_{20K} , f_{100K} , and f_{500K} ; the performance trends observed on this problem are representative of other problems in this class. In Figure 1, we show plots of ρ versus the following PH performance statistics: solution quality, number of iterations required for convergence, and run-time.

As expected, with the noted exception at $\mu = 0$ and $\rho = 100,000$, higher-quality solutions are obtained as the fix lag μ (subject to a fixed ρ strategy) is increased (Figure 1a). However, the highest-quality solutions are not obtained with smaller ρ values. While $\rho = 500,000$ clearly yields lower-quality solutions, there is no clear preference between $\rho = 20,000$ and $\rho = 100,000$. 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 problems supports this general conclusion. Finally, we observe that the total number of PH iterations required for convergence on these large-scale test problems - even using the various acceleration techniques we propose - is quite large, ranging from approximately 50 to over 2000 (Figure 1b).

ρ Selection Strategy	Fix Lag		
	$\mu = 0$	$\mu = 1$	$\mu = 2$
<i>sep</i>	0.1790%	0%	0.0034%
<i>cp</i> _{0.5}	0.4440%	0.1188%	0.3794%
<i>cp</i> _{1.0}	0.4490%	0.3056%	0.0070%
<i>f</i> _{100K}	0.3149%	0.7612%	0.5402%

Table 3: Quality of solutions obtained by variable- ρ and f_{100K} strategies on the $n = 10$, $k = 10$ spares-only test problem; values represent the percentage above the cost of the best known solution.

We now analyze the performance of the variable- ρ strategies, comparing solution quality and run-time relative to both one another and to the f_{100K} baseline strategy; the latter yielded the best overall performance for a fixed- ρ strategy. Solution quality results for the $n = 10$, $k = 10$ test problem, expressed as a percentage above the cost of the best known solution, are shown in Table 3. 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, there is some motivation for identifying a clear winner among the three strategies. Unfortunately, no clear pattern emerges in the analysis of the data in either Table 3 or for the other test problems. 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 problem. Finally, comparing the variable- ρ strategies with the f_{100K} strategy, we find that although f_{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 problems.

Variable- ρ Selection Strategy	Fix Lag		
	$\mu = 0$	$\mu = 1$	$\mu = 2$
<i>sep</i>	375%	456%	312%
<i>cp</i> _{0.5}	198%	342%	152%
<i>cp</i> _{1.0}	52%	178%	122%

Table 4: Run-time of variable- ρ PH strategies on the $n = 10$, $k = 10$ spares-only test problem, relative to the baseline f_{100K} strategy.

Despite superior performance, the solution quality of the variable- ρ strategies does come with an increase in run-time cost. In Table 4, we show the percentage increase in overall run-time for the variable- ρ strategies relative to the f_{100K} strategy on the $n = 10, k = 10$ test problem. 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 are domain-specific heuristics, while the former is general-purpose. 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, 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 problems 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, 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, yielding 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 f_{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 - where 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 f_{100K} selection strategy to obtain low-fidelity solutions.

Num. Bases (n)	Num. Scenarios (k)	% 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 5: 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.

We conclude our analysis of the spares-only test problems by considering the quality of the solutions resulting from PH relative to the absolute baseline costs reported in Table 1, as opposed to the solution costs obtained by other ρ selection strategies. In Table 5, 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 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, PH - even with convergence acceleration techniques - can serve as a very effective heuristic for solving stochastic integer programs.

4.4 Spares Plus Resources Performance Results

As discussed in Section 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 for the SESP. For example, consider the case with $n = 5$ and $k = 10$. As recorded in Table 1, 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 and the empirical observation that additional consideration of resource optimization significantly complicates planning-oriented MIPs, including the SESP. The growth in difficulty with increases in n and k is also remarkable, e.g., CPLEX ran for over 9 days attempting to solve the LP relaxation of the $n = 10$, $k = 10$ 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 problems up to dimension $n = 5$ and $k = 10$. On these problems, 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 problems (reported in Table 5), there is evidence that the increase is primarily due to the weakening of the LP bound when 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 problems.

To avoid replication of the presentation in Section 4.3, we simply note that the main experimental conclusions observed for our PH variants on spares-only test problems extend to spares-plus-resources test problems. 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 n and k , a specific fixed- ρ strategy may 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 tractable (in a practical sense) PH variants. Finally, the domain-specific $cp_{0.5}$ and $cp_{1.0}$ variable- ρ strategies yielded roughly equivalent performance in terms of solution quality relative to the problem-independent *sep* counterpart, 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 case where $n = 10$ and $k = 10$, under a $cp_{0.5}$ 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 problem 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. Substituting the *sep* strategy for the

$cp_{0.5}$ strategy, we observe an increase from 185 to 541 iterations required for convergence, with an increase from 276.10 minutes to 1048.22 minutes. In this case, the growth in difficulty is additionally due to the increase in the number of PH iterations, which is frequently correlated with the number of decision variables. In general, the aggregate PH run-time for spares-plus-resources problems is within a factor of four (and often much less) 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.

4.5 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 4.3 and 4.4 range from several hours to several days, e.g., on the larger $n = 10$, $k = 30$ test problem. Fortunately, as observed by a number of researchers, PH is trivially parallelized by distributing the solution of scenario sub-problems across distinct processors [3, 20]. 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 is the number of CPUs. Consequently, given a modest cluster possessing approximately 50 compute nodes, it is possible to reduce the wall clock solution times of PH on these data sets to at most an hour. Similar deployment platforms are required to achieve tractable optimization via PH on the JSP data sets.

5 Conclusions and Directions for Further Research

We have introduced an important real-world stochastic integer problem and 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 a good value 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 global ρ strategies that are commonly associated with PH implementations reported in the literature, at the expense of slight increases in run-time.

Some of the enhancements that we introduce exploit the fact that we are solving a diet problem 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 general constraints.

Our experimental analysis represents, to the best of our knowledge, the largest and most complex stochastic integer programming problem to which Progressive Hedging has been successfully applied. The algorithmic techniques 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 programming

solvers. In terms of domain-specific impact, our approach to solving the SESP is tractable for both spares-only and spares-plus-resources problem instances. In contrast, nearly all literature on the SESP focuses exclusively on the spares-only case. We are currently in the process of extending the approach to real-world SESP other than JSF.

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] H. Maarten and M.H. van der Vlerk, *Stochastic Integer Programming Bibliography*, <http://mally.eco.rug.nl/biblio/stoprog.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] 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.
- [19] C.C. Sherbrooke, "METRIC: A Multi-Echelon Technique for Recoverable Item Control", *Operations Research*, Volume 16 (1968), pp. 121-141.
- [20] 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.
- [21] 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.
- [22] F.M. Slay and R.M. King, "Prototype Aircraft Sustainability Model", Logistics Management Institute, Washington, D.C. Report AF601-R2 (1987).
- [23] 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.
- [24] D.L. Woodruff and E. Zemel, "Hashing Vectors for Tabu Search", *Annals of Operations Research*, Volume 41 (1993), 123-137.

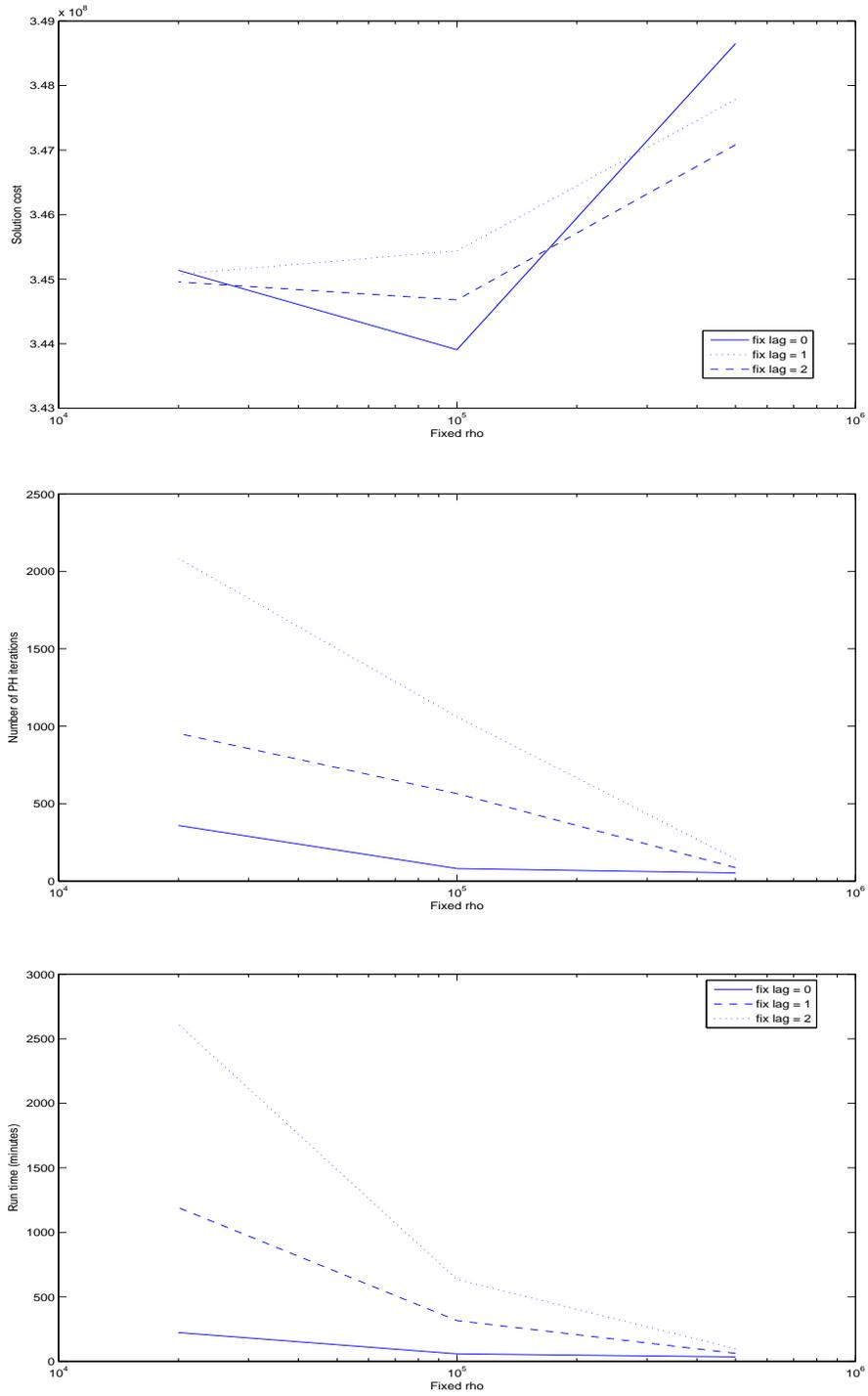


Figure 1: The performance of fixed- ρ PH variants on the $n = 10$, $k = 10$ spares-only test problem. The log-linear plots in the upper, middle, and lower figures display ρ versus (a) solution quality, (b) the number of PH iterations, and (c) run-time, respectively.