12-2023

Utilization of Integer Programming for Scheduling Maintenance at Nuclear Power Plants

Timothy Gallacher
University of Tennessee, Knoxville, tgallach@vols.utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Timothy Gallacher entitled "Utilization of Integer Programming for Scheduling Maintenance at Nuclear Power Plants." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Industrial Engineering.

James Ostrowski, Major Professor

We have read this dissertation and recommend its acceptance:

James Ostrowski, Jamie Coble, Anahita Khojandi, Hugh Medal

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Utilization of Integer Programming for Scheduling Maintenance at Nuclear Power Plants

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Timothy Virgil James Gallacher
December 2023

DEDICATION

I dedicate this work to my children, Roland and Ainslee. If I can instill one thing in each of you, let it be the drive to seek satisfaction in your own hard work.

ACKNOWLEDGEMENTS

First and foremost, I would like to voice my appreciation to my mentor, Professor James Ostrowski. Jim, your guidance has been amazing throughout this entire process, and your patience with me seems to have no bounds. I would also like to thank Professors Anahita Khojandi, Jamie Coble, and Hugh Medal for serving on my dissertation committee, and for all their support and advice throughout my experience at UTK.

I would like to express my appreciation to the University of Tennessee, the Department of Industrial Engineering, and the DOE-NE NEUP, for providing funding for my graduate studies and the work in this dissertation.

I would also like to thank my family for their support and encouragement over the course of this endeavor. Every one of you has backed me in some way, from childcare to lending an ear when I need to vent, and this would not have been possible without your support.

Lastly, I'd like to thank my friends and colleagues for their interest and support these last few years. Bill Levis, Rachel Heath, Kris Strausser: you all have taken a keen interest in my personal and professional development, including this project, which has meant the world to me. My fellow Shift Managers, Bill Sokso, Jeff Weaver, Bill Bulafka, and Casey Coyle: you've been more than accommodating to my needs for shift coverage over the years, and have kept an interest in my on-going work evidenced by your thoughtful questions during our turnovers. Finally, to Ethan Deakins, who has been my closest compatriot during this time of learning and growth. Thank you for your friendship and support in all areas of this endeavor.

ABSTRACT

This thesis develops a thought that naturally explores three specific motifs for solving the complexities of scheduling maintenance at Nuclear Power Plants (NPP). The first chapter of this paper will develop the initial thought around creating a schedule for a given work week, including all the various constraints inherent to this problem. Such constraints include but are not limited to personnel availability, allowable component out-of-service time, and the Plant Risk Assessment. The objective function being to minimize the total cost of worker's compensation for that given week.

The second chapter addresses the question of whether this simple schedule can be implemented with a long time horizon as the goal. This section delves into the concept of utilizing maintenance task frequencies and extended preventive maintenance frequencies to once again minimize the objective function of cost due to compensation.

The third chapter focuses on the ability of the program to respond to adaptive circumstances. One major obstacle in running any large commercial facility is unplanned downtime of required systems or components. Simulating failures of certain components that shorten the overall allowable out-of-service time, the program will be required to still minimize the objective function while navigating these changing timelines.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

The nuclear industry is at a crossroads in its history. Operation and maintenance (O&M) costs contribute 60-70% of nuclear power plants (NPP)s production costs while fuel only contributes 15-20% (Coble, et al. 2015). The additional costs of running a facility post Fukushima, coupled with the increasingly competitive costs of coal and natural gas generated electricity have made nuclear energy non-competitive in many energy markets, leading to premature plant retirements. The necessity for innovation in nuclear energy is crucial as the industry shifts to newer plant technologies such as Small Modular Reactors (IAEA, Advances in small modular reactor technology developments 2020), Advanced Generation IV Reactors (Guiberteau 2020), and potentially fusion reactors on the far horizon (IAEA 2021). As this shift occurs, funding and support for innovative technologies will focus on ways to increase efficiency in the areas of outage scope and duration, thermal power output, and reduction of staffing through maintenance reduction initiatives or automation, to name a few.

One area where innovation can yield benefits and cost savings on multiple fronts is in the realm of scheduling automation. The potential benefits here include reduction in maintenance personnel requirements, increases in efficiency of planners and schedulers, and stricter compliance with NRC Technical Specification and Probabilistic Risk Assessment (PRA) models. This area for innovation is unique, as where most areas will require justification for their increase in efficiency with reductions in safety or quality, an automation of scheduling that integrates with regulatory requirements will inherently maximize the margins to nuclear safety in conjunction with achieving an efficiency increase for all work groups. Additionally, as the industry evolves from a technology perspective, the benefits of automated scheduling endure, ensuring a leaner, optimized

organization no matter the plant design. In addition to reductions in maintenance costs, an automated approach allows for more time to be spent on higher cognitive tasks not suitable for automation.

This can be shown conceptually with the following example. Suppose preventive maintenance on a Residual Heat Removal system requires System Outage Window. Such a component can render a nuclear facility in a Limiting Condition of Operation (LCO) of 72 hours by itself. This requires staffing maintenance personnel around the clock until complete, weeks of planning and refinement of scheduling, multiple series of challenges by Operations and Station Senior Management, all before a single wrench touches system steel. Should a parallel component be emergently out of service at the time of maintenance, this allowable out of service time is drastically reduced, requiring staffing of station leadership around the clock to ensure restoration of all plant components within the LCO timeframe.

CHAPTER I:
AN EXACT METHOD FOR MAINTENANCE SCHEDULE OPTIMIZATION IN NUCLEAR POWER PLANTS

Abstract

This chapter is based on a manuscript prepared for publication by Tim Gallacher, Ethan Deakins, Najmaddin Akhundov, Diego Mandelli, Jamie Coble, Anahita Khojandi, and James Ostrowski:

Akhundov, N., Coble, J., Deakins, E., Gallacher, T., Khojandi, A., Mandelli, D., and Ostrowski, J. (2022). An Exact Method for Maintenance Schedule Optimization in Nuclear Power Plants. *Submitted.* Authors Deakins, Gallacher, and Ostrowski proposed the method. Author Deakins developed the source code. Author Gallacher cross-checked coded results to ensure industry compliance. Authors Deakins and Gallacher drafted the manuscript. Authors Coble, Khojandi, and Ostrowski edited the manuscript.

In this chapter we present an Integer Programming (IP) method for solving a weekly schedule for a High Pressure Injection System (HPIS) developed by the Probabilistic Risk Assessment (PRA) group RAVEN. The goal is to develop an algorithm that minimizes the overall expenditure of O&M funds on personnel wages while completing all required maintenance in a given work week. This program must also maintain compliance with the hypothetical plant's Technical Specifications (TS) and Probabilistic Risk Assessment (PRA).

Nomenclature

Parameters, Indices, and Sets

$i \in I$: All maintenance tasks for the time horizon.

$c \in C_R$: Components with maintenance tasks the system requires in the time horizon.

$c \in C_O$: Components with backlog maintenance tasks. (Common-train components).

$Q \in \varrho$: Sets of components that force the system to only one operable subsystem (train) if components share overlapping downtime.

$t \in T$: Hourly time steps: 1,...,T.

$j \in J$: Maintenance crew types.

$(i, i') \in P$: Maintenance tasks that have a precedence relationship, i.e., task i must precede task i'.

$G \in \mathcal{g}$: Sets of tasks that require the same man-hours, duration, crew type, and preceding and proceeding tasks.

$i \in B_C$: Tasks that bring a component c offline giving it a tag out status upon beginning.

$i \in E_C$: Tasks that remove the tag out status on component c upon completion.

$i \in F_C$: Tasks that designates a component c as operational upon completion.

$i \in A_j$: Maintenance tasks that require maintenance crew type j.

$R \in R$: All sets of minimal cut-sets for the system.

$s \in S$: Allowable shifts for crew members.

$d_t \in D$: The day to which time step t corresponds in the time horizon (MTWRF).

ES_i: Earliest possible start time of task i.

LS_i: Latest possible start time for task i.

H_i: Duration that task i requires for completion.

L_i: Number of crew members task i requires.

M_j: Total number of available crew members of crew type j.

P_{js}: Weekly pay of a crew member of type *j* on shift *s*.

P_C: Projected future cost for completion of backlog maintenance for component *c* in a future week.

$|R|$: Size of set *R*.

$|Q|$: Size of set *Q*.

Variables:

$u_i(t)$: Assumes the value 1 if maintenance task *i* starts at time step *t*, 0 otherwise.

$v_i(t)$: Assumes the value 1 if maintenance task *i* is being performed during time step *t*, 0 otherwise.

$U_C(t)$: Assumes the value 1 if component *c* receives a tag out status at time step *t*, 0 otherwise.

$V_C(t)$: Assumes the value 1 if component *c* has a tag out status at time step *t*, 0 otherwise.

$W_C(t)$: Assumes the value 1 if component *c* has its tag out status removed at time step *t*, 0 otherwise.

X_C: Assumes the value 1 if backlogged component *c* is scheduled for maintenance, 0 otherwise.

$Y_C(t)$: Assumes the value 1 if a component *c* is offline at time step *t*, 0 otherwise.

$Z_C(t)$: Assumes the value 1 after if all maintenance for a component *c* is complete at time step *t*, 0 otherwise.

$\delta i(t)$: Represents the number of crew members assigned to task *i* at time step *t*.

γ_{js}^{dt}: Represents the number of crew members of type *j* working an 8-hour shift *s* at time step *t* corresponding to day *d*.

Γ_{js}: Represents the total number of crew members of type *j* working a 5-day schedule on shift *s*.

λ_{js}^{dt} : Represents the number of crew members of type *j* working a 12-hour shift s at time step *t* corresponding to day *d*.

Λ_{js}: Represents the total number of crew members of type *j* working a 3-day schedule on shift *s*.

$\alpha(t)$: Assumes value 1 if the system has less two operable subsystems at time step *t*.

Introduction

Maintenance Scheduling Problem:

Maintenance research is largely the result of military and industry interest due to its application in their process and systems as a cost reducing factor (McCall 1965). The first analytical studies of maintenance originate in the 1950s and 1960s as many large companies began to use large-scale preventive maintenance plans to reduce failures and unplanned downtime events (Decker 1996). The maintenance scheduling problem is one result of continual growth in maintenance optimization methods from the 1960s and on. It might be easiest to envision the maintenance scheduling problem as a run of the mill job-shop scheduling problem where jobs are the maintenance orders and machines are the maintenance personnel required to perform the orders; however, (Paz and Leigh 1994) note that maintenance scheduling has more intricacies. The maintenance scheduling problem seeks to assign maintenance orders to a sequence and/or the necessary personnel required. This problem arises in many industries such as power generation (Dopazo and Merrill 1975), aircraft (Sriram and Haghani 2003), and rail (Peng, et al. 2011). Generally, these problems are difficult to solve and as the reader will note in the next section, many researches choose to explore these problems using suboptimal methods.

Maintenance Scheduling Models and Solution Techniques:

Many different modeling strategies see use in the effort to capture the details of a system and plan its maintenance tasks. (Berrichi, et al. 2010) propose a bi-objective model to schedule preventive maintenance and production on multiple parallel machines. They seek to balance the trade-offs of production and maintenance by using the bi-objective approach to modeling. They propose an algorithm based on Multi-Objective Ant Colony Optimization (MOACO) and show that it outperforms two well-known Multi-Objective Genetic Algorithms (MOGA). In (Zhong, et

al. 2018), the authors provide a multi-objective non-linear constrained model for scheduling preventive maintenance on offshore wind farms. Their two objectives are to maximize the reliability of the system (based on given reliability criterion) and to minimize the maintenance cost of the system. They employ a MOGA, namely NSGA-II, and provide a set of Pareto-optimal solutions to support decision making in a numerical example.

In (Samaranayake and Kiridena 2012), the authors approach the heavy aircraft maintenance problem by proposing a unitary structuring framework that consolidates the ideas of materials resource planning (MRP) in production planning, critical path method (CPM) for project management, and production activity control (PAC) for shop-floor scheduling. They derive a numerical example from another case study and discuss its effectiveness in the heavy aircraft maintenance planning problem. The authors of (Froger, et al. 2016) discuss maintenance planning/scheduling in the electricity industry. They breakdown maintenance problems into five categories: maintenance planning for generating units, transmission maintenance scheduling and network considerations, management of uncertainty, fuel management, and maintenance scheduling. Further they discuss the range of solution procedures such as genetic algorithms (GA), mathematical programming such as mixed integer linear programming (MILP), simulated annealing (SA), heuristics, particle swarm optimization (PSO), game theory, etc.

(Perez-Canto and Rubio-Romero, 2013) provide a stochastic MILP (SMILP) formulation for the power plant production maintenance scheduling (PPPMS) problem that considers a network of power plants and the optimal scheduling of maintenance for generators within the network. They validate their model on an example instance that comprises 95 generators (17 wind, 20 hydroelectric, 50 thermal, and 8 nuclear) with three demand scenarios by providing an optimal,

valid solution to the problem. The authors of (Mollahassani-Pour, Abdollahi and Rashidinejad 2014) provide a similar model to (Perez-Canto and Rubio-Romero, 2013), but introduce a new method of considering cost reductions for maintenance scheduling using a new metric generated analyzing unit commitment for the generators before solving the maintenance planning problem. (Canto 2008) also models the problem of maintaining power generation units by developing a SMILP that considers scheduling maintenance while solving the unit commitment problem for the generators, as well. The author chooses to use Bender's decomposition to decompose and solve the problem where the master problem schedules maintenance and the subproblem schedules unit commitment per demand period.

Maintenance Scheduling in NPPs:

Scheduling maintenance in NPPs is difficult because the systems within the plant are highly reliable with much component interdependence. Further, safety is of the upmost importance, and as such, high reliability is not just a technical requirement but a safety requirement. In (Lapa, Pereira and Mol 2000), the authors choose to use a GA to solve a scheduling problem for auxiliary feed water system (AFWS). They compare their GA solution to maintenance policies with high maintenance frequency, low maintenance frequency, and no maintenance. (Ayoobian and Mohsendokht 2016) consider multiple decision criteria that couple with a genetic algorithm to produce Pareto-optimal solutions that reduce unavailability, cost, and exposure time (ET). They use allowable outage time (AOT) and PM intervals as decision variables. By using a sensitivity index (SI) to rank the priori- ties of the different decision criteria, they show a reduction of 86%, 58%, and 30% of unavailability, cost, and ET for a simplified high pressure injection system

(HPIS), respectively. The comparison is with respect to initial optimal values obtained for their formulation without SI ranking of the decision criteria.

(Zhang, et al. 2019) analyze a feed water system for a multi-unit NPP. They use a GA tailored to minimize unavailability, cost, and risk. A total of four scenarios (three with certain system performance characteristics, and one without) that use different weights for different types of risk are studied to validate their modeling approach. The result is a general optimization framework for a multi-unit NPP system with uncertainties. (Nilsson, et al. 2009) use a simple integer program to model opportunistic maintenance during maintenance scheduling for a feed water shaft system in an NPP. However, the focus of this study is on opportunistic maintenance strategies rather than general maintenance scheduling. Furthermore, the system they analyze comprises three pumps in the feed water system, but they only consider two at a time, and hence reliability is not a concern in their model.

(Carlos, et al. 2012) use PSO to analyze a motor-driven pump and a HPIS. Their model minimizes cost and unavailability by choosing when to schedule maintenance (as intervals) for the components of the system. They compare their Pareto-optimal solutions to initial schedules obtained from plant data for their systems (initial schedules were the result of manufacturer suggested maintenance intervals). (Harunuzzaman and Aldemir 1996) give a dynamic programming (DP) approach to scheduling maintenance on a few different, simplified NPP systems. They analyze their scheme on two different examples, one having more strict assumptions, and the other relaxing these assumptions to evaluate their approach from a more robust perspective. They show that their framework produces lower cost maintenance solutions to

the problems they examine with the presence of economies of scale, and that by slightly relaxing system requirements it can save large amounts of cost for maintenance without effecting reliability.

Two-Tier Framework

The research in this paper encompasses a two-tier optimization framework that produces optimal cost maintenance schedules and identifies reliability concerns with each new incumbent solution. The two-tier framework contains an integer programming (IP) model that is responsible for producing an optimal cost maintenance schedule and a PRA model to analyze incumbent scheduling solutions for reliability cut-set violations. A reliability cut-set is a set of components that if there is an overlap in all of their downtime, then the reliability of the system falls below a designated threshold. Hence, a violation is any schedule where all components of any cut-set have overlapping downtime. For a comprehensive guide into RAVEN and its capabilities, see (Alfonsi, et al. 2021). When and if cut-set violations occur, cutting planes are generated ad hoc to remove these cut-set violations from the next incumbent solution. Figure 1 illustrates the core integration of the IP and the RAVEN PRA models into a capable scheduling program.

The IP model in this study seeks to minimize the weekly labor cost of maintenance crew members while scheduling maintenance activities that the system requires in an upcoming week. The model is subject to labor allocation constraints, technical specification constraints, and logical constraints modeling the behavior of components while crew members perform these components' maintenance. Further, the model attempts to schedule maintenance for components that are on backlog. Scheduling these backlog components credits a projected future cost to the NPP in the objective function.

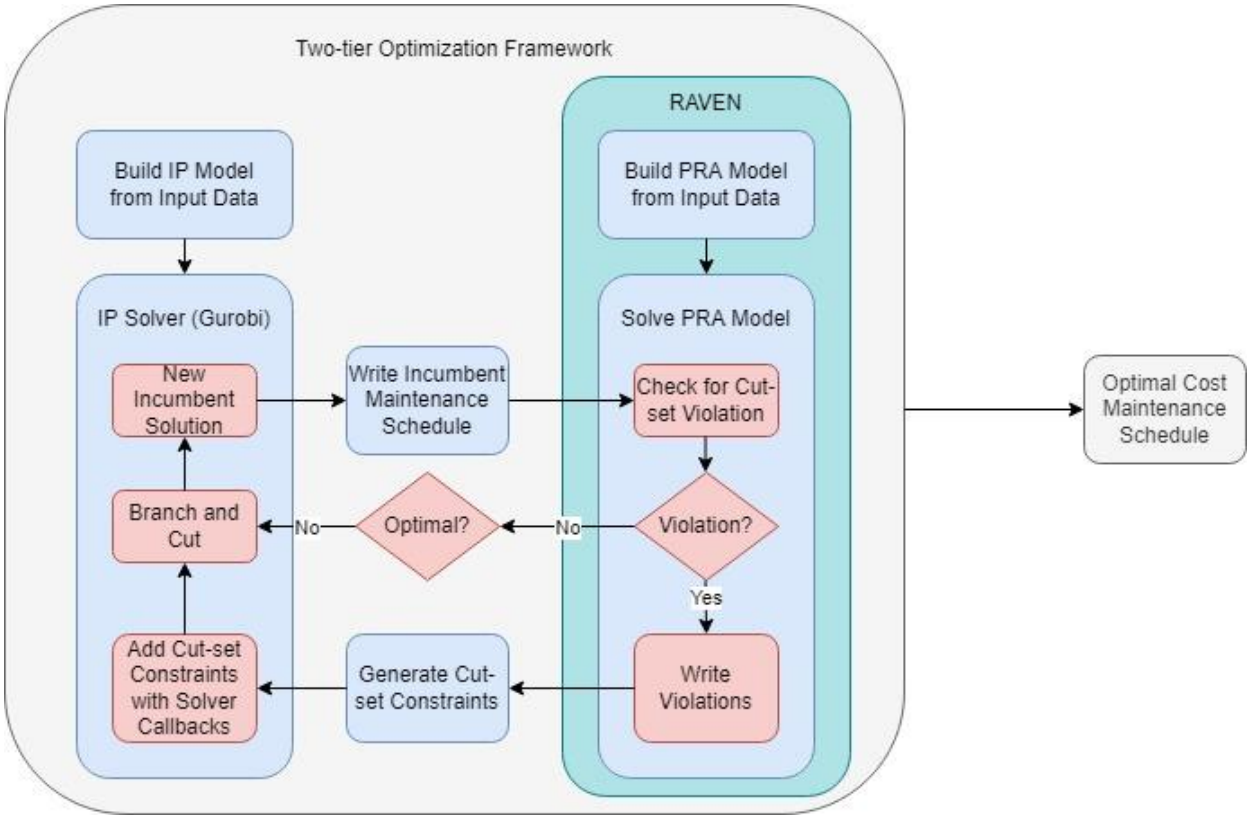

Figure 1: Two-Tier Optimization Framework

From a high-level perspective, we provide input data describing the system and maintenance tasks associated with the system to the two-tier framework. The two-tier framework then generates an IP model instance and a PRA model instance. The two-tier framework solves the IP model by way of branch and cut. For each new incumbent solution to the IP model, the two-tier framework generates a maintenance schedule corresponding to the current incumbent solution and passes the schedule to RAVEN to evaluate with the PRA model. RAVEN solves the PRA model and identifies whether any cut-set violations exist in the schedule. When violations exist, RAVEN records the cut-sets and passes them back to the main two-tier framework for cut-set constraint generation. Once the two-tier framework generates all necessary cut-set constraints, the IP solver accepts the constraints as input by means of a callback utility. The callback utility adds each cut-set constraints before branch and cut continues which cuts off the current solution with cut-set violations. The two-tier framework concludes in an optimal cost maintenance schedule with no reliability cut-set violations.

Integer Programming Model

The IP model in this study seeks to minimize the weekly labor cost of maintenance crew members while scheduling maintenance activities that the system requires in an upcoming week. The model is subject to labor allocation constraints, technical specification constraints, and logical constraints modeling the behavior of components while crew members perform these components' maintenance. Further, the model attempts to schedule maintenance for components that are on backlog. Scheduling these backlog components credits a projected future cost to the NPP in the objective function.

Model:

Minimize: $$\sum_{j \in J} \sum_{s \in S} P_{js}(\Gamma_{js} + \Lambda_{js}) \quad (1)$$

Subject to:

$$\sum_{t = ES_i}^{LS_i} u_i(t) = 1 \qquad \forall i \in I(c), \forall c \in C_R \quad (2)$$

$$\sum_{t = ES_i}^{LS_i} u_i(t) = X_c \qquad \forall i \in I(c), \forall c \in C_O \quad (3)$$

$$\sum_{t\prime = t - H_i + 1}^{t} u_i(t') = v_i(t) \qquad \forall t \in T, \forall i \in I \quad (4)$$

$$L_i v_i(t) = \delta_i(t) \qquad \forall t \in T, \forall i \in I \quad (5)$$

$$\sum_{t \in T} \delta_i(t) \leq \gamma_{js}^{dt} + \lambda_{js}^{dt} \qquad \forall j \in J, \forall i \in A_j, \forall s \in S \quad (6)$$

$$\gamma_{js}^{dt} \leq \Gamma_{js} \qquad \forall j \in J, \forall s \in S, \forall d_t \in D \quad (7)$$

$$\sum_{dt \in D} \gamma_{js}^{dt} = 5\,\Gamma_{js} \qquad \forall j \in J, \forall s \in S \quad (8)$$

$$\lambda_{js}^{dt} \leq \Lambda_{js} \qquad \forall j \in J, \forall s \in S, \forall d_t \in D \quad (9)$$

$$\sum_{dt \in D} \lambda_{js}^{dt} = 3\,\Lambda_{js} \qquad \forall j \in J, \forall s \in S \quad (10)$$

$$\sum_{s \in S} \Gamma_{js} = \Lambda_{js} \leq M_j \qquad \forall j \in J \quad (11)$$

$$\sum_{t\prime = t}^{LS_i} u_i(t') + \sum_{t\prime = ES_{i\prime}}^{t + H_i - 1} u_{i\prime}(t') \leq 1 \qquad \forall (i, i') \in P, \forall t \in T \quad (12)$$

$$u_i(t) = U_c(t) \qquad \forall t \in T, \forall i \in B_c, \forall c \in C_R \cup C_O \quad (13)$$

$$u_i(t - H_i) = W_c(t) \qquad \forall t \in T, \forall i \in E_c, \forall c \in C_R \cup C_O \quad (14)$$

$$u_i(t - H_i) = Z_c(t) \qquad \forall t \in T, \forall i \in F_c, \forall c \in C_R \cup C_O \quad (15)$$

$$U_c(t) - W_c(t) = V_c(t) - V_c(t - 1) \qquad \forall t \in T, \forall c \in C_R \cup C_O \quad (16)$$

$$U_c(t) - Z_c(t) = Y_c(t) - Y_c(t - 1) \qquad \forall t \in T, \forall c \in C_R \cup C_O \quad (17)$$

$$\sum_{t = ES_i}^{LS_i} v_i(t) = D_i \qquad \forall i \in I(c), \forall c \in C_R \quad (18)$$

$$\sum_{t=ES_i}^{ES_i} v_i(t) = D_i X_c \qquad \forall i \in I(c), \forall c \in C_O \qquad (19)$$

$$\sum_{t=ES_i}^{LS_i} t \cdot u_i(t) \leq \sum_{t=ES_{i'}}^{LS_{i'}} t' \cdot u_{i'}(t) \qquad \forall (i, i') \in G \qquad (20)$$

$$\sum_{c \in C_R} \sum_{i \in B_c} u_i(0) = 1 \qquad (21)$$

$$\sum_{c \in Q} V_c(t) \leq (|Q| - 1) + \alpha(t) \qquad \forall t \in T, \forall Q \in q \qquad (22)$$

$$\sum_{t'=t}^{t+72} \alpha(t') \leq 72 \qquad \forall t \in T \qquad (23)$$

$$\sum_{c \in R} V_c(t) \leq |R| - 1 \qquad \forall t \in T, \forall r \in R \qquad (24)$$

$$u_i(t), v_i(t), U_c(t), V_c(t), W_c(t), X_c, \alpha(t) \in \{0, 1\} \qquad (25)$$

$$\gamma_{js}^{dt}, \lambda_{js}^{dt}, \Gamma_{js}, \Lambda_{js} \in \{0, 1, 2, \ldots, \infty\} \qquad (26)$$

The objective in (1) minimizes the cost of labor and is a departure from the original model where the projected future cost of backlog maintenance is no longer credited. This is because the back log maintenance is now replaced with emergent maintenance, which must be performed within the required timeframe of 72 hours. Constraint (2) ensure that each maintenance task the system requires begins in the time interval [ES_i, LSi]. Constraint (3) ensures that if a backlog maintenance component c appears in the maintenance schedule, then all maintenance tasks *i* for that component begin in the time interval [ES_i, LS_i]. Constraint (4) ensures that a task completes in consecutive time steps. Constraint (5) requires that the correct number of maintenance crew members to complete any task found in the scheduling solution. Constraint (6) restricts the number of crew members the model assigns at any time step t to no more than the number of crew members the model assigns to the day and shift corresponding to that time step. Constraint (7) restricts the number of crew members the model assigns to an 8-hour shift *s* on any day d_t to no more than the

total number of crew members the model assigns to a 5-day week on shift *s*. Constraint (8) ensures each crew member the model assigns to an 8-hour shift *s* works five of those shifts in the week. Constraints (9) and (10) perform the same function of the previous two constraints except for the crew members that the model assigns to a 3-day work week of 12-hour shifts. Constraint (11) ensures the total number of crew members the model assigns to five 8-hour shifts and three 12-hour shifts is no more than the available number of crew members within the NPP for each crew type.

Constraint (12) restricts the starting time step of a task *i'* to be after task *i* is complete when these two tasks have a precedence relationship $i \prec i'$. Each component the model schedules for maintenance has both a task that brings that component offline and a task that brings that component online. In industry this is referred to as placing and removing a tagout or lockout on these components. Constraint (13) ensures that a component is brought offline and receives a tagout at the same time step *t* that its tagout task occurs. Constraint (14) removes a component's tagout at the same time step that its tagout removal task is complete. Constraint (15) designates a component to be available from a PRA perspective when the final maintenance task for that component is complete, including the tagout removal. Constraint (16) ensures that a component retains its tagout until the task that removes the tagout status is complete. Constraint (17) ensures a component remains offline until the final maintenance task for that component is complete. Constraints (18) and (19) help to tighten the model for all maintenance tasks the model schedules. Constraint (20) helps to break the symmetry of scheduling maintenance tasks that require the same man-hours, duration, crew-type, and preceding/proceeding tasks. Constraint (21) ensures that one of the components that has maintenance tasks the system requires is brought offline at time step 1.

Constraint (22) forces the value of $\alpha(t)$ if the maintenance schedule moves the system into a 72-hour LCO window. Specifically concerning the system in this paper, $\alpha(t)$ assumes the value 1 when only one of the three system trains is operable per the simulated system licensing basis. In this case, the system can remain in this state for no longer than 72 hours (time steps) and the model enforces this with Constraint (23). We list Constraint (24) in the formal description of the model; however it is actually a cutting plane constraint that the two-tier framework adds to the model at each new incumbent solution if a cut-set violation occurs in that solution.

Implementation

The main programming language for the two-tier framework is Python. Referring to Figure 1, the outermost rectangle can be thought of as a python driver program that constructs the IP model with the PYOMO modeling language, writes any data necessary for the program, and calls RAVEN to perform PRA analysis. The Python programming language presents itself as the most advantageous to this study as PYOMO modeling language and RAVEN have a Python foundation. The following paragraphs break down the two-tier framework to its individual blocks and describe the process for building that specific piece of the program.

First, we construct the IP model section of Figure 1 with the PYOMO modeling language. PYOMO is a complete python implementation of a powerful optimization modeling language that can be used with any open-source or commercial solver that has the capability to read an “lp” or “mps” file. For more information on the PYOMO modeling language, see (Hart, et al. 2017). We create functions in the driver program to construct the PYOMO model that is the result input data that describes maintenance tasks and the system of interest. All input data are read from a spread sheet file. A complete description of the input data is in Section V. The two-tier framework

constructs the model using Python functions and PYOMO modeling language techniques and passes it to Gurobi, a commercial optimization solver.

Next, the main driver code controls the RAVEN portion of Figure 1.1. In this study, we use a RAVEN built-in cut-set solver to perform PRA. We construct the driver in such a way that it easily communicates scheduling solutions with RAVEN. RAVEN is a powerful reliability analysis tool and is capable of reading data from "csv" files. With this, the two-tier framework generates "csv" files at each new incumbent IP solution that encode the scheduling information and are passes this csv file as input to RAVEN.

Finally, the driver code integrates the PYOMO model and the RAVEN model by writ- ing schedules from the IP solution and reading cut-set information from RAVEN. The two-tier framework calls RAVEN as a sub-process at each new incumbent solution by executing a callback function within Gurobi. Most modern, commercial solvers allow users to define callback functions that evaluate and perform certain functions given the status of the solver. Once, a new incumbent is found, Gurobi uses the callback function to access the internal solver values for the variables and write them as a maintenance schedule to an exterior "csv" file. The same callback function then invokes RAVEN to read that "csv" file and analyze it for cut-set violations. RAVEN also requires an "xml" file that it interprets to construct the cut-set solver model and identify to report its findings. Once RAVEN reports its findings, the original callback function parses the RAVEN output and uses any violated cut-sets to generate constraints and then add them to the model. Once this is complete, Gurobi continues the branch-and-cut process solving the IP model with the additional cut-set constraints

Tests and Results

Data:

Our point-of-contact in industry allows for us to collect previous maintenance records that describe regular maintenance tasks within their NPP. We remove certain descriptors from data in all listings of this paper as it would reveal the NPP that employs our point-of-contact which is a security risk. Likewise, we cannot provide schematics for an actual injection system within the point-of-contact's NPP. With this in mind, we choose to make use of a schematic for a hypothetical HPIS included in (Alfonsi, et al. 2021) and map maintenance data onto this system.

The HPIS divides into three trains that describe the three paths that coolant can be injected from the RWST to the RPV. In practice, NPP maintenance scheduling typically looks to schedule maintenance that isolates only a single train for a given week. This extends to similarly labeled trains in additional safety related systems or divisions of Class 1E electrical power. This approach helps to reduce the likelihood of reliability concerns such as cut-set violations that result in a total system shut-down, which would also drive the NPP to commence a reactor shut down.

We denoted three trains in the HPIS in Figure 2 by A, B, and C and they contain the components (P1, V3), (P2, V6, V7), and (P3, V5), respectively. We consider components V1, V2, and V4 to be Common-train components since one may insert them into work windows involving two or more of trains above. They are also considered common as they are involved in the physical flow path for more than one discrete subsystem. Disruption of flow through one of these common components impacts the functionality of two or more subsystem.

Finally, this study does not consider RPV and RWST as any maintenance on these components requires the entire system to be brought offline. Work on scheduling of these types

of activities can be considered for future research where such schedule modeling is implemented to refueling or maintenance outages with the reactor offline.

Since our HPIS system is not actually found in the NPP that provided the maintenance task data, we must choose a method for mapping maintenance tasks onto the system we use in this paper. Through consultation with our point-of-contact, we select tasks from the maintenance data to perform on the hypothetical HPIS as the components in the system are actual components one might find in a NPP. Using this method, we map a maintenance sequence (grouping of tasks required to properly maintain a component) onto each component in the system excluding the RWST and RPV. Full listings of maintenance data for each component are in the Chapter 1 Appendix.

A sample excerpt of maintenance tasks for component P1 in A-train is in Table 1. The first column describes the task activity. The second column assigns a task identification number to each task. The third column places a flag on specific tasks that apply a tag out to a component or remove a tag out from a component. A value of 0 refers to a task that gives a component a tag out when that task begins and a value of 1 refers to a task that removes a tag out from a component when that task is complete. The fourth column describes the time the NPP allots to that task for completion and the fifth column describes the man-hours a task requires to finish in the time the NPP allots. Dividing the fifth column values by the corresponding fourth column values gives the number of crew members required for a task. The sixth column lists the crew type qualified to complete a task. The seventh column describes precedence relationships between tasks where the value gives the tasks that must precede the task in the corresponding row.

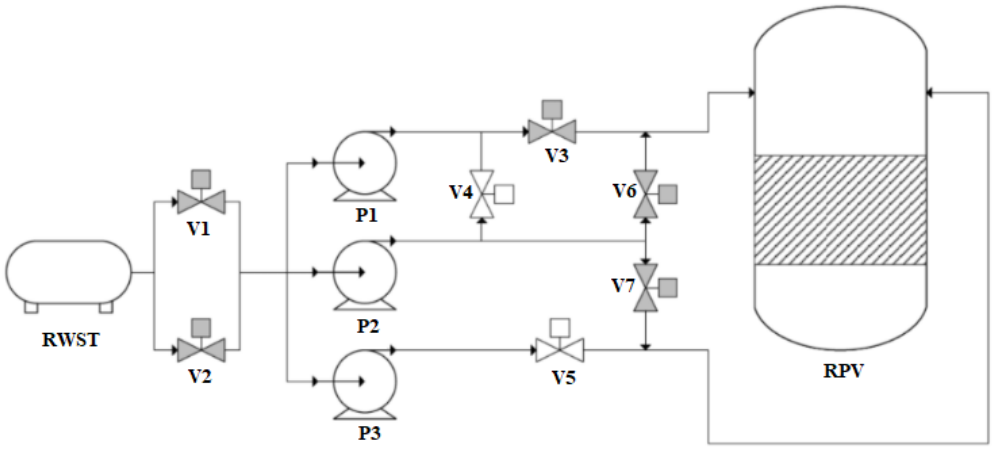

Figure 2: HPIS

Table 1: Sample Input Data for A-Train HPIS Components

A-train Tasks	Task Number	Hang/Remove Tag Out	Duration (Hours)	Man-Hours	Crew	Precedence
Hang P1 Tagout	0	0	3	6	SSV2	-1
De-term P1 Motor	1	–	6	24	LMM5	0
Uncouple P1 Pump/Motor	2	–	6	24	LMM3	0
Clean/Inspect Motor	3	–	8	32	LMM5	1
Flush P1 Motor Oil	4	–	4	16	LMM5	3
Add P1 Motor Oil	5	–	2	4	LMM5	4
P1 Bus Relay/Instrument Cals	6	–	8	24	LMI1	1
P1 Tan-Delta Testing	7	–	8	32	LMM5	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Remove P1 Tagout	18	1	3	6	SSV2	16 17

Test Setup and Results:

From an optimization perspective, the input maintenance data for A-train and C-train are the same. We test and validate the two-tier framework only using A and B-train maintenance task data for this reason. The model has a limited number of crew available to assign to shifts and tasks. The available crew values are in Table 2. We assume all crew members, regardless of type, are paid at a rate of $24/hour and $28.50/hour for day and evening/night shifts, respectively. The total labor cost of the schedule in Figure 3 is $28,920. We construct three different test scenarios for each train's maintenance data. The first scenario does not allow the model to schedule any backlog maintenance tasks. The second scenario gives the model complete control over whether to schedule backlog maintenance tasks. Finally, the last scenario forces the model to schedule backlog maintenance for at least one Common-train component. Completing all maintenance tasks for a Common-train component results in a $5,000 premium that the model credits to the NPP in the objective function.

Tables 3 and 4 describe the two-tier framework's performance under each scenario for A-train and B-train. That is, they list the best objective and best relative/actual gap after 30 minutes of solving, as well as the best known objective, final gap, and final solve time for each instance. Column one describes the test scenario. Columns two and three show the best known objective value and best relative and actual gap after 30 minutes of solver time, respectively. The actual gap is the percent difference between the optimal objective value and the best known value at 30 minutes of solver time. Column four shows the best known objective value at the two-tier framework time limit. Column five shows the final optimality gap. Lastly, column six shows the final run-time for the two-tier framework in each scenario.

Table 2: Available Crew Members

Crew Type	SSV2	SSV3	LMM3	LMM5	LMM6	LMI1
# Available	8	4	8	8	8	3

Table 3 shows that the two-tier framework obtains an exact optimal solution for the A-train maintenance workload when not considering backlog tasks. Further, the two-tier framework obtains the optimal solution by 30 minutes into the solve and utilizes the rest of the solve time to verify optimality. Conversely, in the last two scenarios the two-tier framework does not obtain an exact optimal solution within the four-hour time limit. In Table 4, no instance solves to exact optimality within the four-hour time limit.

The last two scenarios for A-train and B-train have equal best known objectives at the two-tier framework time limit, respectively. This is a function of the premium value the model credits to the NPP when backlog maintenance completes. That is, if we reduce the premium, it is possible that when the NPP performs backlog maintenance it increases the objective value as compared to the option to not perform backlogged maintenance. This produces a situation in which the model will exclude backlog maintenance. However, the NPP may choose to force backlog maintenance due to other requirements or timelines within the plant which is why we validate the two-tier framework under both scenarios. Note that if backlog maintenance is feasible and financially advantageous then the final objective values for the last two scenarios in any maintenance workload will be the same. The reasoning here is that given feasibility and a financial advantage to backlog maintenance, we may remove the constraint that forces at least one backlog maintenance component's maintenance as the model will choose to schedule one in the optimal solution. Thus, the two models are the equivalent.

Tables 5 and 6 show an optimal scheduling schema for maintenance crew members when model solves for the A-train data set with no backlog maintenance.

Table 3: A-Train Cost and Timing

A-train					
Backlog Maintenance	Best Obj (1800 secs)	Best Relative/ Actual Gap (1800 secs, %)	Best Obj	Final Gap (%)	Final Run-time (secs)
None	$28,920.00	18.10/0.00	$28,920.00	0.00	13,000.26
Model Decides	$23,480.00	43.30/–	$22.760.00	36.40	14,400.00
At Least One	$22,760.00	42.80/–	$22,760.00	39.28	14,400.00

Table 4: B-Train Cost and Timing

B-train					
Backlog Maintenance	Best Obj (1800 secs)	Best Relative/ Actual Gap (1800 secs, %)	Best Obj	Final Gap (%)	Final Run Time (secs)
None	$30,840.00	16.60/–	$38,040.00	10.89	14,400.00
Model Decides	$27,760.00	29.90/–	$27,760.00	28.17	14,400.00
At Least One	$27,760.00	29.30/–	$27,760.00	27.09	14,400.00

Table 5: Optimal Crew Schedule for A-Train with No Backlogged Maintenance (8-hour Shifts)

	Monday		Tuesday		Wendesday		Thursday		Friday	
Crew	7:00 - 15:00	15:00 - 23:00	7:00 - 15:00	15:00 - 23:00	7:00 - 15:00	15:00 - 23:00	7:00 - 15:00	15:00 - 23:00	7:00 - 15:00	15:00 - 23:00
SSV2	–	–	–	–	–	–	–	–	–	–
LMI1	2	–	2	–	2	–	2	–	2	–
LMM3	4	–	4	–	4	–	4	–	4	–
LMM5	–	–	–	–	–	–	–	–	–	–
LMM6	–	–	–	–	–	–	–	–	–	–

Table 6: Optimal Crew Schedule for A-Train with No Backlogged Maintenance (12-hour Shifts)

	Monday		Tuesday		Wendesday		Thursday		Friday	
Crew	7:00 - 19:00	19:00 - 7:00	7:00 - 19:00	19:00 - 7:00	7:00 - 19:00	19:00 - 7:00	7:00 - 19:00	19:00 - 7:00	7:00 - 19:00	19:00 - 7:00
SSV2	–	–	–	–	–	–	–	–	–	–
LMI1	1	–	1	–	1	–	–	–	–	–
LMM3	–	–	–	–	–	–	–	–	–	–
LMM5	4	–	6	–	4	–	4	–	–	–
LMM6	4	–	4	–	4	–	–	–	–	–

The schedules shown Figures 3 and 4 are seen to be quite different from each other. Aside from the entry into the maintenance window, and exit from the window, the order and timing of the tasks shows little to no overlap. The differences here can be ascribed to a primary cause with secondary implications, with the saving of time being the primary driver to the legacy model scheduler. The Optimized schedule will preferentially plan work on normal 8-hour workdays, Monday through Friday, in order to not incur overtime costs for working evening/night shifts. This accounts for the large gaps in activity between clusters of workflow. The legacy scheduler on the other hand, will normally assign a day and night-shift crew to expedite the LCO window. This is not optimal from a cost-savings perspective; however, it does ensure that the LCO does not expire with the maintenance window still active.

This is an area within the industry there is an opportunity for improvement with respect to the expenditure of O&M funds. The benchmark industry operators have procedural guidelines in place for scheduling of certain duration LCO windows. For certain operating stations, a 7-day LCO such as this would require around-the-clock coverage, however it can be clearly shown in Figure 3 that such expenditure of resources is not a requirement. An adoption of optimal cost maintenance schedules will respect the bounds of the LCO, while simultaneously seeking to reduce overtime costs wherever possible. In this particular case, the two-tier framework schedules all maintenance that the system requires within 120 hours, which is well within the 7-day or 168-hour allowance. The cost of the optimal A-train maintenance schedule without backlog maintenance is $28,920.00. The schedule prepared by hand for the A-train maintenance workload results in a cost of $36,612.00. The optimal schedule reduces the cost of labor by 21%.

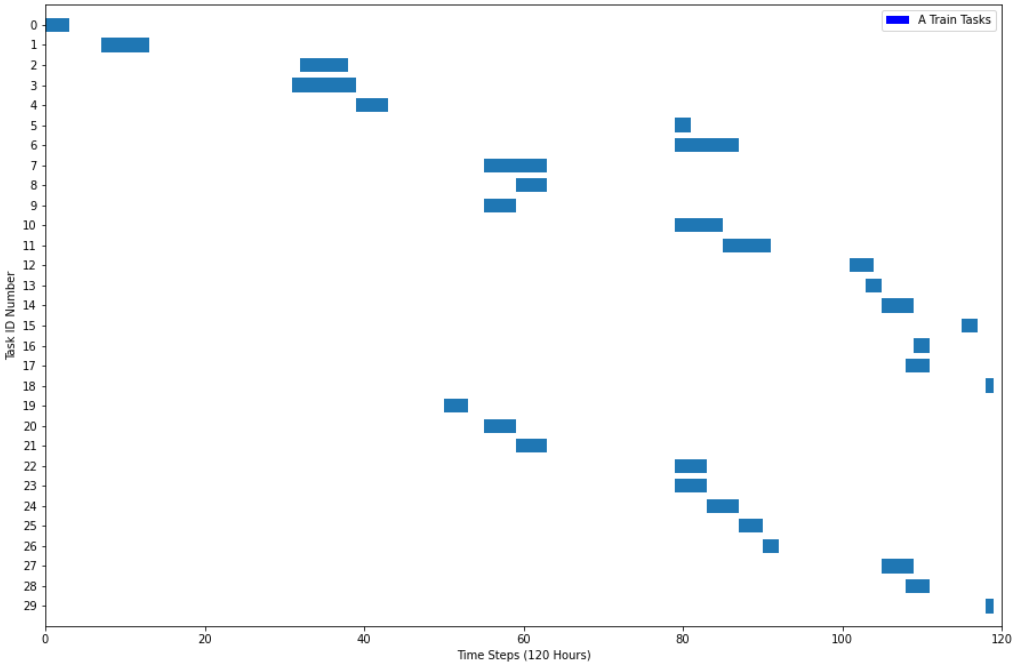

Figure 3: Optimal A-Train Maintenance Schedule with No Backlogged Maintenance

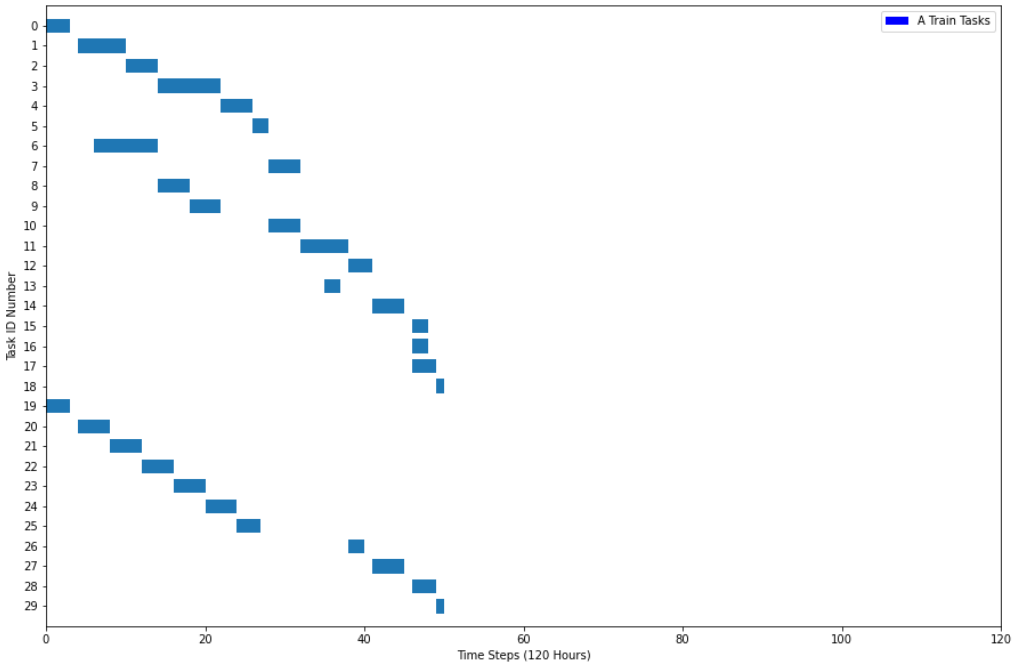

Figure 4: Point of Contact Hand Constructed Schedule

Figure 5 shows the optimal maintenance schedule for the A-train subsystem of the HPIS when the two-tier framework considers backlog maintenance. The two-tier framework incorporates all backlog maintenance tasks into a schedule that simultaneously contains all tasks that A-train requires. This reduces the objective value below the baseline of only scheduling tasks that A- train requires. Note that there are more than 41 tasks listed in the plot; however, the tasks are differentiated by whether they are backlog or not and have overlapping indices. Red bars in the plot show backlog maintenance task time frames and blue blocks show task time frames for maintenance tasks that A-train requires. There is some overlap in backlog tasks and tasks that A-train requires as evident in Figure 5. The labor cost of this schedule is $27,760. Figures 6 and 7 in Appendix B depict the maintenance schedules for the best known maintenance scheduling solution without and with backlog maintenance, respectively.

Discussion

This study presents a two-tier framework for modeling and solving maintenance scheduling problems in NPPs. We construct an integrated approach to handle an IP optimization model that couples with a PRA model to ensure optimal cost, reliable maintenance schedules. This two-tier framework allows RAVEN to identify cut-set violations as they occur and the IP model to add constraints to remove these cut-set violations via a callback function. We construct data sets for this study by mapping real-world NPP maintenance data onto a hypothetical HPIS that might be found in a NPP. We collect this real-world data from previous maintenance workload data sets in the NPP that our point-of-contact is employed by. We remove all plant identifiers from the data sets for security purposes.

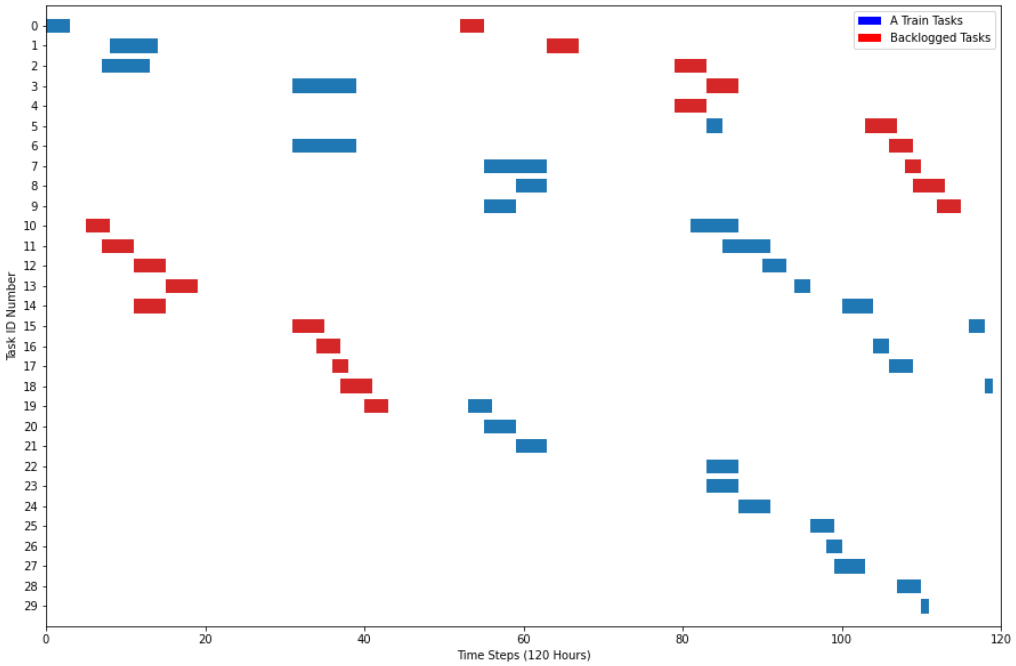

Figure 5: Best Known A-Train Maintenance Schedule with Backlogged Maintenance

We validate the two-tier framework by using it to solve the maintenance scheduling problem for two major trains of the HPIS using three scenarios for each train. Further, we compare an optimal schedule resulting from the two-tier framework against a hand-generated schedule in accordance with the current maintenance scheduling techniques used by our point-of-contact in industry. This paper establishes that there is reason to continue further research in automated methods of maintenance scheduling within NPPs. The cost savings shown is this chapter are likely an overestimate as no outside systems are considered in the maintenance scheduling process. For a true assessment of the cost savings that could be achieved, the two-tier framework should incorporate an entire maintenance workload for all systems undergoing maintenance in a given week. However, this paper shows promising early results for this two-tier framework and provides a foundation for further research into using exact methods for optimal maintenance scheduling in NPPs.

APPENDIX

Table 7: Complete List of A-Train Maintenance Tasks

A-train Components	A-train Tasks	Task Number	Hang/Remove Tagout	Duration (Hours)	Man-Hours	Crew	Precedence
P1	Hang P1 Tagout	0	1	3	6	SSV2	-1
P1	De-term P1 Motor	1		6	24	LMM5	0
P1	Uncouple P1 Pump/Motor	2		6	24	LMM3	0
P1	Clean/Inspect Motor	3		8	32	LMM5	1
P1	Flush P1 Motor Oil	4		4	16	LMM5	3
P1	Add P1 Motor Oil	5		2	4	LMM5	4
P1	P1 Bus Relay/ Instrument Cals	6		8	24	LMI1	1
P1	P1 Tan-Delta Testing	7		8	32	LMM5	1
P1	Impeller visual Inspection	8		4	16	LMM3	2
P1	Repack P1 Packing	9		4	8	LMM3	2
P1	Re-Couple P1 Pump/Motor	10		6	24	LMM3	8 9
P1	Re-term P1 Motor	11		6	24	LMM5	5 7
P1	Remove P1 Tagout	12	0	3	6	SSV2	10 11
P1	P1 Test Setup	13		2	4	SSV2	12
P1	P1 Fill/Vent	14		4	8	SSV2	13
P1	P1 PMT - Flow Test	15		2	8	SSV2	14
P1	P1 PMT - No Leaks at Sys Pressure	16		2	6	SSV2 LMM3	14
P1	P1 PMT - Bus Trip Testing	17		3	6	LMI1	14
P1	P1 OPERABLE	18		1	2	SSV2	15 16 17
V3	Hang V3 Tagout	19	1	3	6	SSV2	-1
V3	Limitorque Removal	20		4	16	LMM6	19
V3	V3 Internal Inspection/ Cleaning	21		4	16	LMM6	20
V3	V3 Stem Cleaning	22		4	8	LMM6	21
V3	V3 Repack	23		4	8	LMM6	20
V3	V3 Reassembly	24		4	16	LMM6	22 23
V3	Remove V3 Tagout	25	0	3	6	SSV2	24
V3	V3 Test Setup	26		2	4	SSV2	25
V3	V3 Fill/Vent	27		4	8	SSV2	26
V3	V3 Diagnostic Testing	28		3	12	LMM6	27
V3	V3 & P1 OPERABLE	29		1	2	SSV2	28

Table 8: Complete List of B-Train Maintenance Tasks

B-train Components	B-train Tasks	Task Number	Hang/Remove Tagout	Duration (Hours)	Man-Hours	Crew	Precedence
P2	Hang P2 Tagout	0	1	3	6	SSV2	-1
P2	De-term P2 Motor	1		6	24	LMM5	0
P2	Uncouple P2 Pump/Motor	2		6	24	LMM3	0
P2	Clean/Inspect Motor	3		8	32	LMM5	1
P2	Flush P2 Motor Oil	4		4	16	LMM5	3
P2	Add P2 Motor Oil	5		2	4	LMM5	4
P2	P2 Bus Relay/ Instrument Cals	6		8	24	LMI1	1
P2	P2 Tan-Delta Testing	7		8	32	LMM5	1
P2	Impeller visual Inspection	8		4	16	LMM3	2
P2	Repack P2 Packing	9		4	8	LMM3	2
P2	Re-Couple P2 Pump/Motor	10		6	24	LMM3	8 9
P2	Re-term P2 Motor	11		6	24	LMM5	5 7
P2	Remove P2 Tagout	12	0	3	6	SSV2	10 11
P2	P2 Test Setup	13		2	4	SSV2	12
P2	P2 Fill/Vent	14		4	8	SSV2	13
P2	P2 PMT - Flow Test	15		2	8	SSV2	14
P2	P2 PMT - No Leaks at Sys Pressure	16		2	6	SSV2 LMM3	14
P2	P2 PMT - Bus Trip Testing	17		3	6	LMI1	14
P2	P2 OPERABLE	18		1	2	SSV2	15 16 17
V6	Hang V6 Tagout	19	1	3	6	SSV2	-1
V6	Limitorque Removal	20		4	16	LMM6	19
V6	V6 Internal Inspection/ Cleaning	21		4	16	LMM6	20
V6	V6 Stem Cleaning	22		4	8	LMM6	21
V6	V6 Repack	23		4	8	LMM6	20
V6	V6 Reassembly	24		4	16	LMM6	22 23
V6	Remove V6 Tagout	25	0	3	6	SSV2	24
V6	V6 Test Setup	26		2	4	SSV2	25
V6	V6 Fill/Vent	27		4	8	SSV2	26
V6	V6 Diagnostic Testing	28		3	12	LMM6	27
V6	V6 & P2 OPERABLE	29		1	2	SSV2	28
V7	Hang V7 Tagout	30	1	3	6	SSV2	-1
V7	Limitorque Removal	31		4	16	LMM6	30
V7	V7 Internal Inspection/ Cleaning	32		4	16	LMM6	31
V7	V7 Stem Cleaning	33		4	8	LMM6	32
V7	V7 Repack	34		4	8	LMM6	31
V7	V7 Reassembly	35		4	16	LMM6	33 34
V7	Remove V7 Tagout	36	0	3	6	SSV2	35
V7	V7 Test Setup	37		2	4	SSV2	36
V7	V7 Fill/Vent	38	24	4	8	SSV2	37
V7	V7 Diagnostic Testing	39		3	12	LMM6	38
V7	V7 & P2 OPERABLE	40		1	2	SSV2	39

Table 9: Complete List of Common-Train Maintenance Tasks

Common-train Components	Common-train Tasks	Task Number	Hang/Remove Tagout	Duration (Hours)	Man-Hours	Crew	Precedence
V1	Hang V1 Tagout	0	1	3	6	SSV2	-1
V1	Limitorque Removal	1		4	16	LMM6	0
V1	V1 Internal Inspection/ Cleaning	2		4	16	LMM6	1
V1	V1 Stem Cleaning	3		4	8	LMM6	2
V1	V1 Repack	4		4	8	LMM6	1
V1	V1 Reassembly	5		4	16	LMM6	3 4
V1	Remove V1 Tagout	6	0	3	6	SSV2	5
V1	V1 Test Setup	7		2	4	SSV2	6
V1	V1 Fill/Vent	8		4	8	SSV2	7
V1	V1 Diagnostic Testing	9		3	12	LMM6	8
V2	Hang V2 Tagout	10	1	3	6	SSV2	-1
V2	Limitorque Removal	11		4	16	LMM6	10
V2	V2 Internal Inspection/ Cleaning	12		4	16	LMM6	11
V2	V2 Stem Cleaning	13		4	8	LMM6	12
V2	V2 Repack	14		4	8	LMM6	11
V2	V2 Reassembly	15		4	16	LMM6	13 14
V2	Remove V2 Tagout	16	0	3	6	SSV2	15
V2	V2 Test Setup	17		2	4	SSV2	16
V2	V2 Fill/Vent	18		4	8	SSV2	17
V2	V2 Diagnostic Testing	19		3	12	LMM6	18
V4	Hang V4 Tagout	20	1	3	6	SSV2	-1
V4	Limitorque Removal	21		4	16	LMM6	20
V4	V4 Internal Inspection/ Cleaning	22		4	16	LMM6	21
V4	V4 Stem Cleaning	23		4	8	LMM6	22
V4	V4 Repack	24		4	8	LMM6	21
V4	V4 Reassembly	25		4	16	LMM6	23 24
V4	Remove V4 Tagout	26	0	3	6	SSV2	25
V4	V4 Test Setup	27		2	4	SSV2	26
V4	V4 Fill/Vent	28		4	8	SSV2	27
V4	V4 Diagnostic Testing	29		3	12	LMM6	28

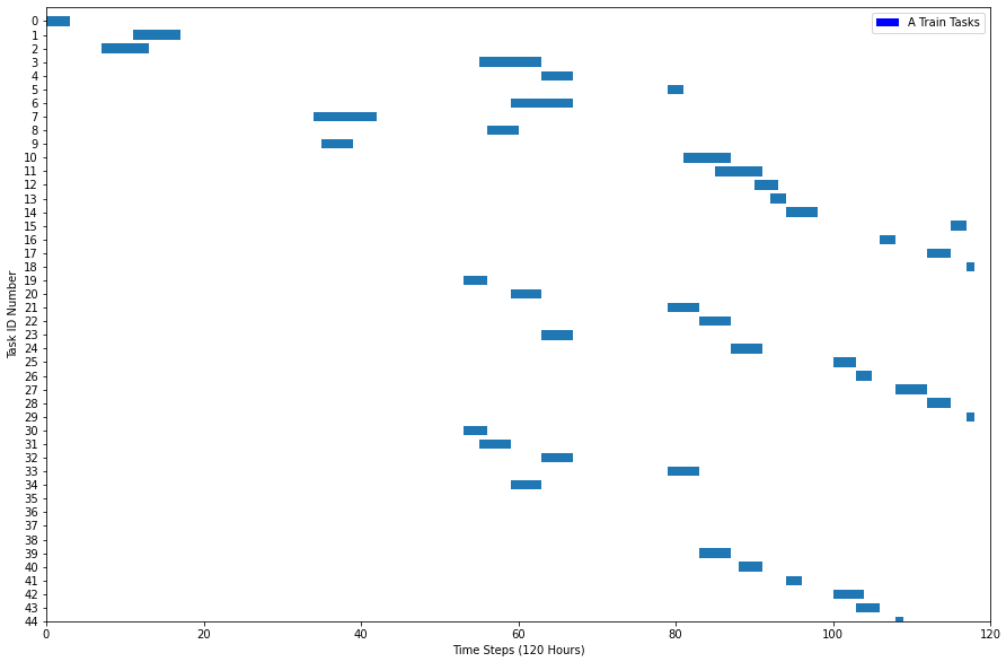

Figure 6: Optimal B-Train Maintenance Schedule Without Backlogged Maintenance

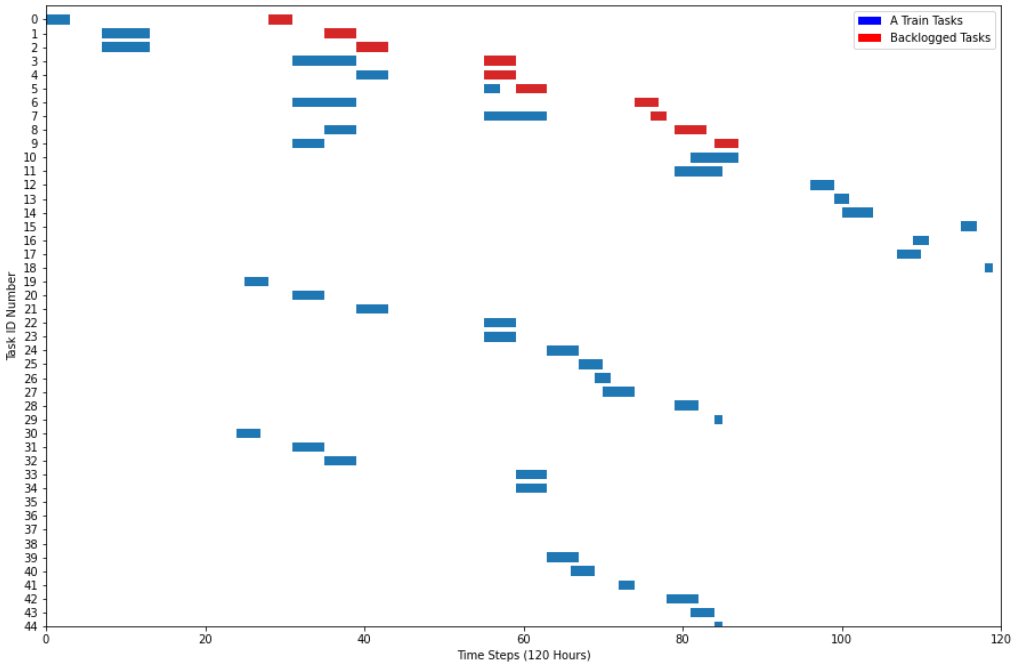

Figure 7: Optimal B-Train Maintenance Schedule With Backlogged Maintenance

CHAPTER II
LONG-TERM MAINTENANCE PLANNING IN NUCLEAR POWER PLANTS VIA INTEGER PROGRAMMING AND DECOMPOSITION

Abstract

Building on the work of Deakins et al, this paper analyzes a more complex set of problems inherent to scheduling maintenance activities at industrial nuclear facilities. Previous work sought to lay the groundwork for an integer programming solution utilizing multiple work groups and maintenance tasks on a single train of a nuclear safety system. This solution sought to minimize the cost of labor subject to a series of constraints that model the regulatory requirements of removing required safety systems from service to perform maintenance. While important to establish a use case for the methodology, the original work requires expanding upon in order to be fully useful in a commercial setting.

To further develop the technology and explore what discreet issues may arise during its implementation, the team expanded the maintenance task list of the NPP safety injection system to include all trains and components, with each task being repeated on a 3-week periodicity. Furthermore, components that were considered common between all or multiple trains were set to repeat on a 4-week periodicity to provide some variability for the IP, as well as simulate the concept of "grace period" with respect to NPP scheduling variability. A 9-week schedule was then created by the IP to show the tasks reoccurring over an extended period of time, and how the variability of the common-train items can be modeled to continue reducing labor costs.

Adjustments to the back end of the model were also made and will be documented in this paper to show what approximations or reductions may be appropriate in this setting. These adjustments are performed in the interest of computational performance, as the problem of a 9-week schedule is orders of magnitude higher than that of a single week. Continual improvement of computational times will also be instrumental in any subsequent work on larger time horizons,

more complex systems, or emergent schedule disruptions. A summary of the changes will comprise an appendix of the complete paper, and may be of higher interest to industrial engineers and programmers than the raw output of the IP itself.

Lastly, as with the original paper, a compare and contrast segment will be dedicated to showing the cost savings of utilizing an IP solution such as ours, as well as the time savings on the part of the scheduler. This breakdown will be instrumental in describing to industry leaders the benefits of the methodology in terms of cost and time savings.

Nomenclature

Parameters:

$|\cdot|$: Set size operator.

$w \in W$: Weeks in the Time Horizon.

$w \in W_s$: Weeks that require maintenance on subsystem *s*.

$s \in S$: Subsystems of the plant being modeled.

$c \in C$: Components not belonging to a specific subsystem (i.e., Common-train).

$f_c \in \mathcal{F}$: Required maintenance frequency for component $c \in C$.

$I \in \Im$: All combinations of $c \in C, \forall n \in |C|$.

$P^w_{(s,I)}$: Maintenance cost for maintaining subsystem *s* and components *i* during week *w*.

Variables:

$X^w_{s,I}$: 1 if subsystem *s* and common-train components in set *I* maintenance completes in week *w*, 0 otherwise.

Introduction

In the U.S., operation and maintenance (O&M) costs account for ap- proximately 60-70% of the total operating cost of nuclear power plants (Coble, et al. 2015). Many efforts have been made to optimize scheduling maintenance actions for individual assets to minimize the maintenance cost, subject to required safety considerations. However, a holistic approach is more advantageous as it allows plant-wide, or even multi-site, maintenance planning. A holistic framework must consider long-term maintenance planning, as well as more granular short-term maintenance scheduling with safety and reliability requirements.

Long-term Maintenance Planning in NPPs

Maintenance scheduling of standby safety systems has much interest in the literature. The authors of (Harunuzzaman and Aldemir 1996) demonstrate a Dynamic Programming (DP) approach to determining the optimal cost schedule for checking, maintenance, repair, and replacement (CMRR) of system components over the course of a year across many scenarios manipulating failure rates and reliability requirements.

In (Martorell, et al. 2002), the authors consider the same HPIS from (Harunuzzaman and Aldemir 1996) but extend the problem to consider technical specifications and maintenance (TS&M) decisions rather than simply maintenance decisions. They formulate a risk-based objective and cost-based objective model and solve both using a steady-state genetic algorithm (SSGA). They demonstrate their approach across different scenarios defined by varying numbers of TS&M decision variables and show significant reductions in cost and risk metrics compared to initial TS&M values for their example system. The study in (Jiejuan, Dingyuan and Dazhi 2004) incorporates TS&M decisions into probability safety analysis (PSA) via a risk-cost maintenance optimization model that examines the effects of optimal cost SIs and PMIs on NPP plant risk. They

solve their risk-cost model using a GA and demonstrate the optimal SIs for a hypothetical 10 component system.

The authors of (Hadavi 2008) take a step back from risk-cost models covered in the previous literature and focus instead on improving the GA approach itself in solving risk-cost models for maintenance optimization. They introduce an evaluation function for their GA that weights three different attributes of the model: risk, maintenance cost, and loss of revenue. Further, they show the effectiveness of their GA routine at finding reasonably good scheduling solutions in only a few thousand iterations for maintaining an auxiliary feedwater system (AFWS) and compare their results to a Monte Carlo simulation. The optimization approach of (Pereira, et al. 2010) changes to PSO to solve a non-periodic maintenance scheduling program focused on minimizing unavailability and maintenance cost for a further simplified version of the HPIS in [2, 3]. The authors demonstrate the feasibility of PSO for solving a PM optimization problem using three scenarios that manipulate the weights on unavailability and cost in the objective function of their model.

The work in (Carlos, et al. 2012) extends on the work of (Pereira, et al. 2010) using PSO to provide a Pareto front for the maintenance scheduling problem minimizing unavailability and maintenance cost when decision variables are uncertain. They validate their approach by solving the maintenance scheduling problem using SI and PM periods as decision criteria on a motor-driven pump and HPIS. The authors use Monte Carlo simulation and order statistics to develop confidence intervals on uncertain decision variables and, thus, uncertain objective function values. The focus on risk-cost models shifts in (Aghaie, et al. 2013) where the authors use an advanced progressive real coded GA (APRCGA) to maximize system availability. They show better system

availability with their method using an AFWS and RHRS as example systems and comparing to other methods from (Lapa, Pereira and Mol 2000).

In (Chou, Ge and Zhang 2014), the authors present a new MOO algorithm with a foundation in PSO. The authors use this new algorithm, denoted as multi-objective based PSO (MOBPSO), to produce a set of nondominated, Pareto-optimal solutions without having to convert their problem to a single-objective optimization model. The authors of (Ayoobian and Mohsendokht 2016) build on the research concerning Pareto-optimal solutions to risk-cost models for NPP maintenance scheduling by introducing a sensitivity index measure for choosing solutions from the Pareto front minimizing unavailability, maintenance cost, and exposure time with decision variables SIs, PMIs, and AOTs. They solve their model using non-dominating sorting GA II (NSGA-II) and show a 86%, 58%, and 30% reduction in unavailability, maintenance cost, and exposure time using their sensitivity index compared to initial values from [3], respectively.

The authors of (Ge, et al. 2018) use coevolutionary multiswarm PSO (CMPSO), pro- posed in (Zhan, et al. 2013), with adaptations to handle mixed integer MOO models. They compare their method to NSGA-II and note the merits of its uniformity performance and that its computational time is comparable to that of NSGA-II. The work in (Zhang, et al. 2019) extends the literature on maintenance scheduling via a GA to solve a MOO maintenance model designed to analyze multi-unit NPPs. They use NSGA-II to minimize multi-unit unavailability, plant risk, and plant cost across three different scenarios of varying concern for off-site in- dividual doses above and below 50 mSv. The research in (Ito and Suzuki 2020) moves away entirely from metaheuristics and uses a mixed integer linear program (MILP) to model NPP maintenance scheduling by minimizing the total number of maintenance activities. The scope of this model is component level maintenance

scheduling over the course of a multi-year time horizon, similar to the other literature cited in this research.

Contributions:

We propose a solution to this holistic framework using a multi-tier frame- work that integrates long and short-term maintenance scheduling via integer programming and model decomposition. A long-term maintenance sched- ule should consider short-term maintenance costs that are evaluated on a week-to-week basis. A short-term maintenance schedule seeks to minimize the weekly maintenance cost while adhering to reliability requirements and technical specifications. The solution to the multi-tier methodology must comprise an optimal cost long-term maintenance schedule for plant components that respects short-term maintenance costs and requirements.

We present a multi-tier framework responsible for producing optimal cost long-term maintenance schedules while considering short-term maintenance costs and factors. We demonstrate our multi-tier framework using a hypothetical HPIS in Figure 8 and two long-term maintenance planning situations. Both situations require a fifty-two week maintenance schedule. Situation 1 places no restrictions on short-term considerations while Situation 2 does. We provide optimal cost maintenance schedules and multi-tier run-time break- downs for both situations.

Methods

Long-term Model

This paper focuses on a multi-tier framework for online, long-term maintenance scheduling in NPPs. The first tier of the methodology is a long-term maintenance scheduling model that seeks to produce an optimal sequence of maintenance for plant or multi-site assets. A typical NPP facility

decomposes plant assets into subsystems of components if possible. With that in mind, the long term model produces an optimal cost sequence of maintenance for plant subsystems that adheres to technical specifications and plant specifications for system maintenance. E.g., consider the hypothetical high pressure injection system (HPIS) in Figure 8. There are three main subsystems or trains in this HPIS that are responsible for coolant flow from the reactor water storage tank (RWST) to the reactor pressure vessel (RPV).

A-train comprises components P1 and V3, B-train comprises components P2, V6, and V7, and C-train comprises components P3 and V5. Components V1, V2, and V4 are designated as Common-train components meaning maintenance on these components can occur alongside any other train as long as the maintenance is within plant and technical specifications. We do not consider RWST and RPV in this paper as these components require offline maintenance. The long-term maintenance model seeks to minimize the cost of scheduling required maintenance on each train including the correct sequence of maintenance for common-train components over a given time horizon.

We assume that plants or multi-site assets can be decomposed into subsystems (even if it is a trivial decomposition) and that these subsystems have a required frequency of maintenance. Further, we assume that maintenance on subsystem maintains its entirely to an acceptable level of functionality. The long-term time horizon can be any amount of time but we classify that time in terms of weeks as to integrate with the short-term maintenance scheduling.

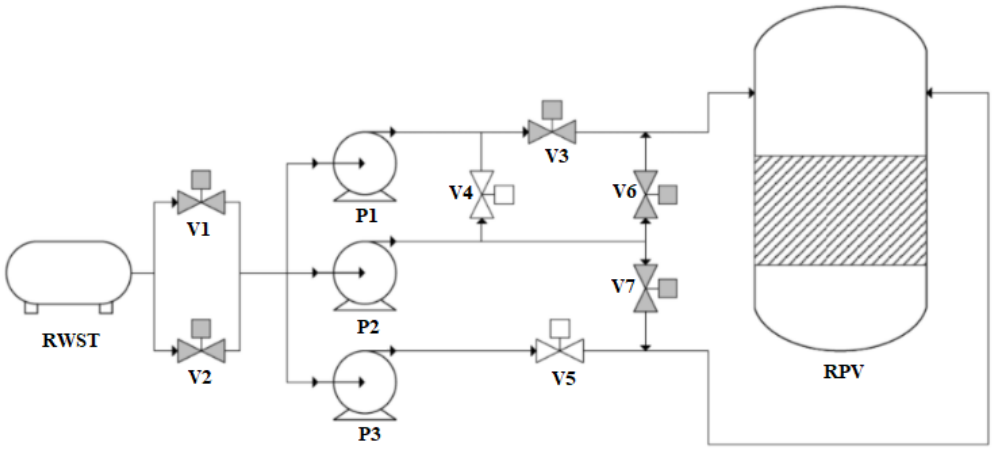

Figure 8: HPIS

Model:

Minimize: $$\sum_{j \in J} \sum_{s \in S} P_{js}(\Gamma_{js} + \Lambda_{js}) \quad (1)$$

Subject to:

$$\sum_{t = ES_i}^{LS_i} u_i(t) = 1 \qquad \forall i \in I(c), \forall c \in C_R \quad (2)$$

$$\sum_{t = ES_i}^{LS_i} u_i(t) = X_c \qquad \forall i \in I(c), \forall c \in C_O \quad (3)$$

$$\sum_{t\prime = t - H_i + 1}^{t} u_i(t') = v_i(t) \qquad \forall t \in T, \forall i \in I \quad (4)$$

The objective function (1) minimizes the total cost of maintenance over the long-term time horizon. Constraint (2) ensures that a component c not belonging to a specific subsystem has maintenance performed at least every f_c weeks. Constraint (3) ensures that asset subsystems have maintenance performed in weeks that require the subsystem to be maintained. Finally, constraint (4) restricts all variables to be binary.

Short-Term Model:

The short-term model is an integer program as well and seeks to schedule individual maintenance tasks for a given subsystem throughout the course of a week. Further, the short-term model schedules individual maintenance tasks for components not belonging to a specific subsystem if they were scheduled for the current week by the long-term model. The cost of maintenance in the short term model correlates to the cost of labor as we assume materials and tools are fixed costs and as such do not include them in the objective function. The time horizon of the short-term model is a week and more specifically 120 hours as this corresponds to Monday, Tuesday, Wednesday, Thursday, and Friday (MTWRF) at twenty-four hours per day. The short-

term maintenance schedule must adhere to restrictions on crew allocation, precedence relationships for tasks, limiting conditions of operation (LCO)s and asset reliability requirements.

We make a few adaptations to the short-term model and solution procedure from (Deakins, et al. 2021) to be integrated with the multi-tier framework in this research. First the short-term model in this research does not consider backlog components and does not credit the NPP for finishing backlog maintenance. Second, the short-term model is solved as an integer program, however may not require an integer solution to improve the long-term maintenance schedule.

Solution Procedure:

The long-term model objective coefficients require objective costs from many different instances of the short-term model. From (Deakins, et al. 2021), solving the short-term model can be quite difficult and require much time. Furthermore, for even a small example problem such as the hypothetical HPIS in Figure 1, many of these short-term instances must be solved. Indeed, the long-term model requires $|\boldsymbol{S}| \times |\boldsymbol{I}|$ short-term instance solves to construct its objective function.

To combat the difficulty and time requirement of solving many short-term model instances, we assume that for a given week the more maintenance that occurs the more expensive the week will be. Further, we assume that more complicated weeks in terms of maintenance workload may have their weekly maintenance cost estimated using less complicated weeks. E.g., refer to Figure 1 and take components P1 and V3 belonging to A-train and components V1, V2, and V4 from the Common-train classification. Each week in the long-term model that requires A-train maintenance has eight possible schedules that complete the required maintenance. The possible schedules are in Table 10.

One could solve each short-term instance associated with the eight possible scenarios above. However, recall that we assume that a schedule with more maintenance is more expensive that one without and with this we can estimate the costs of scenarios with more maintenance. Back to the example above, let $P_{(A)}$, $P_{(A,V\,1)}$, $P_{(A,V\,2)}$, and $P_{(A,V\,4)}$ be the cost of the schedules in scenarios (A), (A,V1), (A,V2), and (A,V2), respectively. Now we can estimate the cost of scenario (A,V1,V2), say $P_{(A,V\,1,V\,2)}$, as

$$P_{(A,V\,1,V\,2)} = P_{(A,V\,1)} + P_{(A,V\,2)} - P_{(A)}.$$

We can continue this method until we have either a real cost or estimated cost for all possible scenarios that satisfy a week requiring A-train maintenance. We use the same strategy to obtain maintenance costs for all $|\boldsymbol{S}| \times |\boldsymbol{I}|$ maintenance scenarios in the long-term model.

With the objective coefficients in hand, the long-term model is a simple assignment problem which allows us to obtain a long-term maintenance schedule quickly. However, in general the estimated costs from above are typically less than the true cost of the short-term model they are associated with. This leads to a situation where the long-term model will favor solutions that rely heavily upon weeks that couple the required subsystem's maintenance and many non-subsystem specific components' maintenance, e.g., a week such as (A, V1, V2, V4). This is an undesirable outcome for the NPP because it increases the likelihood of reliability issues for the week in question. Furthermore, since the cost of the heavier maintenance weeks are estimated, there is no guarantee that the short-term model instance associated with that week is feasible with respect to labor allocation, LCOs, and reliability requirements.

Table 10: Possible Long-Term Solutions for a Week Requiring A-Train Maintenance

Possible A-train Schedules
A
A, V1
A, V2
A, V4
A, V1, V2
A, V1, V4
A, V2, V4
A, V1, V2, V4

We combat this outcome by refining the solutions to the short-term model instances associated with estimated maintenance costs. Given an optimal- cost long-term maintenance solution, we take any week in the solution that corresponds to maintenance with an estimated cost and add these weeks to a pool. For each of these weeks in the pool, we use the short-term model and now solve it for this scenario to improve the linear program (LP) relaxation bound increasing the objective cost of this scenario in the long-term model. Once we update all objective coefficients for the short-term scenarios in the pool, we re-optimize the long-term model and obtain a new long-term maintenance scheduling solution. Figure 9 shows a high level of the multi-tier framework.

Implementation:

We construct our multi-tier framework using Python, PYOMO, Gurobi, and RAVEN. Python is used as the base language for constructing the main driver of the code. We use PYOMO to build both the long-term and short- term models. PYOMO is a complete python implementation of a powerful optimization modeling language that can be used with any open-source or commercial solver that has the capability to read an “lp” or “mps” file. For more information on the PYOMO modeling language, see (Hart, et al. 2017).

Gurobi is a high-performance optimization solver. For details on the Gurobi optimization solver and its capabilities see (Optimization 2022). RAVEN (Risk Analysis Virtual Environment) is a flexible and multi-purpose uncertainty quantification, regression analysis, probabilistic risk assessment, data analysis and model optimization framework developed by Idaho National Laboratory. For a comprehensive guide into RAVEN and its capabilities, see (Alfonsi, et al. 2021) and for its use in the short-term model see (Deakins, et al. 2021).

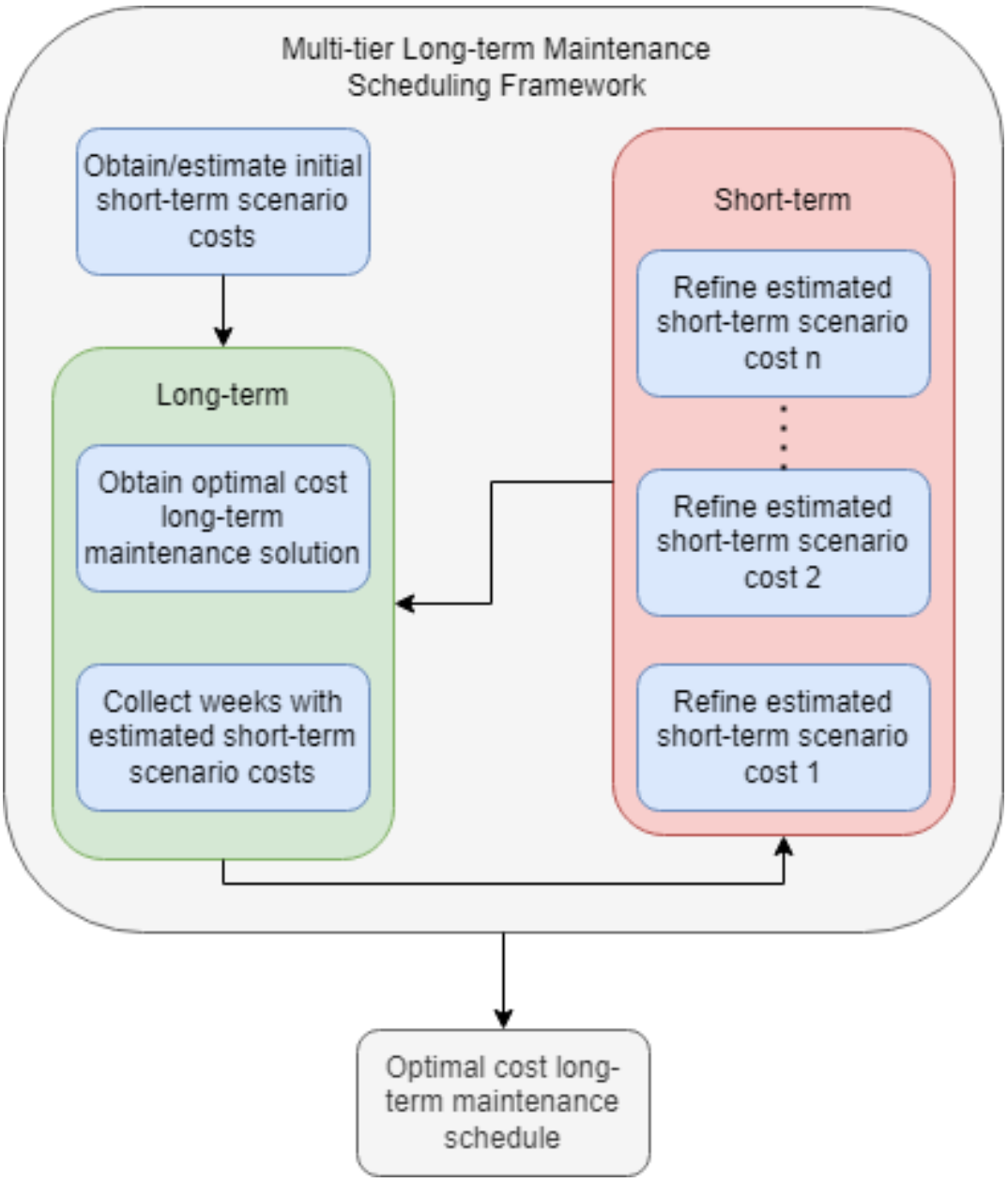

Figure 9: Multi-Tier Framework

We use Python to interface with PYOMO and build both the long-term and short-term models. Python is also used to interface with RAVEN as the software's back-end code is largely Python based as well. Using native Pyomo classes, Gurobi is called to solve instances of both the long and short- term model as needed. Solving the short-term model instances each time they need to be refined to improve the LP relaxation bound is the most time consuming component of the multi-tier framework. To help combat this, we make use of the Python Multiprocessing module that allows for process based parallel programming.

The multi-tier framework executes until weeks in the long-term solution do not have an estimated objective coefficient. However, it is possible to impose other stopping criteria, such as allowing for long-term solutions with estimated cost weeks if the week only includes a certain amount of non- specific subsystem maintenance. E.g., the NPP operating the HPIS in Figure 8 might allow estimated cost weeks for weeks only containing one non-specific subsystems components such as (A, V1) or (C, V4) and so forth. Further, if a week may not contain certain pairs of maintenance work, such as (B, V1, V2), it can be restricted by removing the variable representing that combination in the long-term model.

Results

Data:

Input data for short-term model scenarios comes from the input data in (Deakins, et al. 2021). This short-term model input data includes maintenance tasks, crew requirements, reliability requirements, cut-set probabilities, etc. We construct the input data for the long-term model as follows. Components belonging to the main subsystems of the hypothetical HPIS require maintenance every three weeks. That is maintenance for the A-train subsystem must occur on

weeks {1,4,7,...,51}, maintenance for the B-train subsystem must occur on weeks {2, 5, 8, . . . , 52}, and maintenance for the C-train subsystem must occur on weeks {3, 6, 9, . . . , 49}.

Now, components (V1, V2, V4) require maintenance to be performed at least every four weeks. There is no restriction on which week the maintenance must occur in. Further, maintenance need not be four weeks apart for any one component. That is, given maintenance for any common-train component on week w, then it may re-occur on any week in the range $[w + 1, w + 4]$, and must re-occur by week $w + 4$. Each Common-train components can be paired with any main train subsystem in the hypothetical HPIS unless there are reliability issues, but this is handled in the short-term refinements.

Outcomes:

We validate our two-tier framework using the hypothetical HPIS in Figure 8 and schedule a year long maintenance plan under two different situations. The first situation places no restrictions on short-term maintenance scenarios and allows the long-term model to choose any short-term maintenance scenario as long as it satisfies constraints (2) and (3). The second situation restricts certain short-term scenarios from being schedule in the long-term maintenance solution.

We provide a summary of the performance of the multi-tier framework and the optimal cost long-term maintenance solutions for both situations. The first time any short-term maintenance scenario is refined we impose a 600-second time limit. If, however, the LP relaxation bound cannot be improved and the same short-term scenario appears in another long-term maintenance solution we extend the solve time limit. This continues until the bound can be improved or until the bound is proven to be worse than the estimated cost of that short-term scenario.

Figure 10 shows a fifty-two week optimal cost maintenance schedule for the hypothetical HPIS in Figure 1 under no short-term scenario restrictions. After two short-term refinement stages and three long-term model solves a final optimal solution with no short-term scenarios having estimated costs. The initial long-term cost of this run is $892,380 and the final cost after short-term refinements is $892,920 as seen in Table 12. Table 11 shows the breakdown of run-times for both situations. The run-times for the first situation are in column 1 of Table 11 with a total run-time of 4,173.55 seconds.

Figure 11 shows a fifty-two week optimal cost maintenance schedule for the hypothetical HPIS where the short-term maintenance scenario (B, V1) is not allowed to occur in any week. The initial long-term cost of this schedule $895,080 with a final long-term cost of $895,620. The multi-tier framework run-times for Situation 2 are similar to the run-times for Situation 1 requiring 55.18 seconds more time in terms of total run-time for Situation 2. Further, the objective value for Situation 2 is slightly more expensive than the objective value for Situation 1 and this is expected since Situation 2 restricts the solution space for the long-term model.

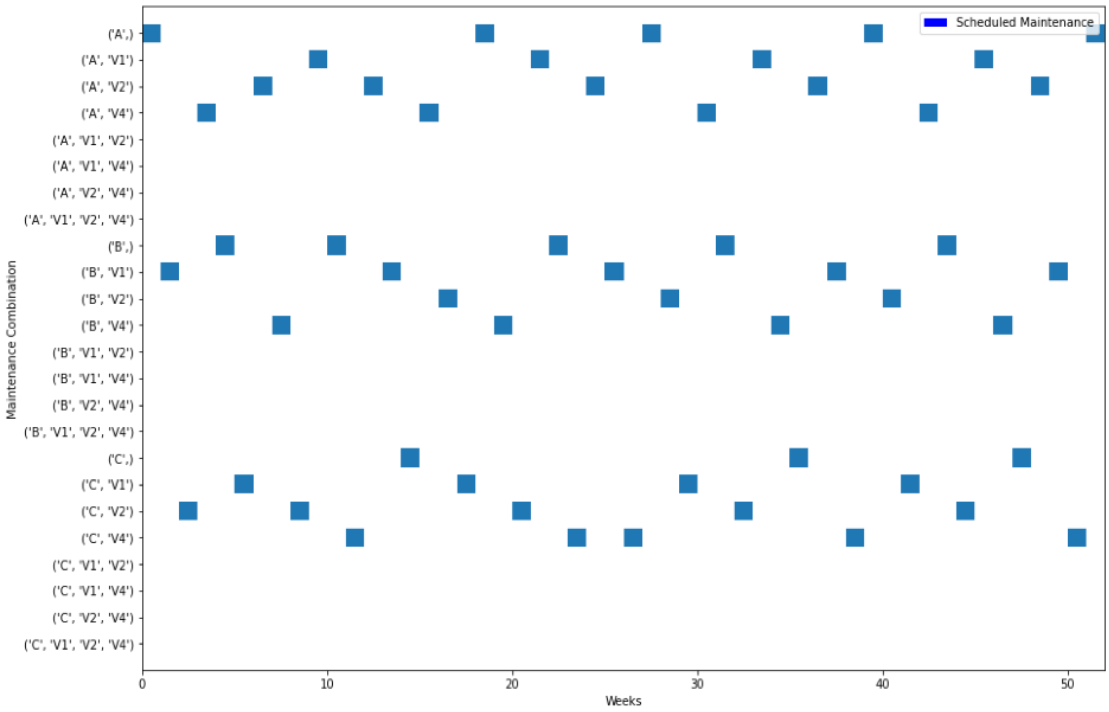

Figure 10: Optimal Long-Term Maintenance Schedule with No Short-Term Scenario Restrictions

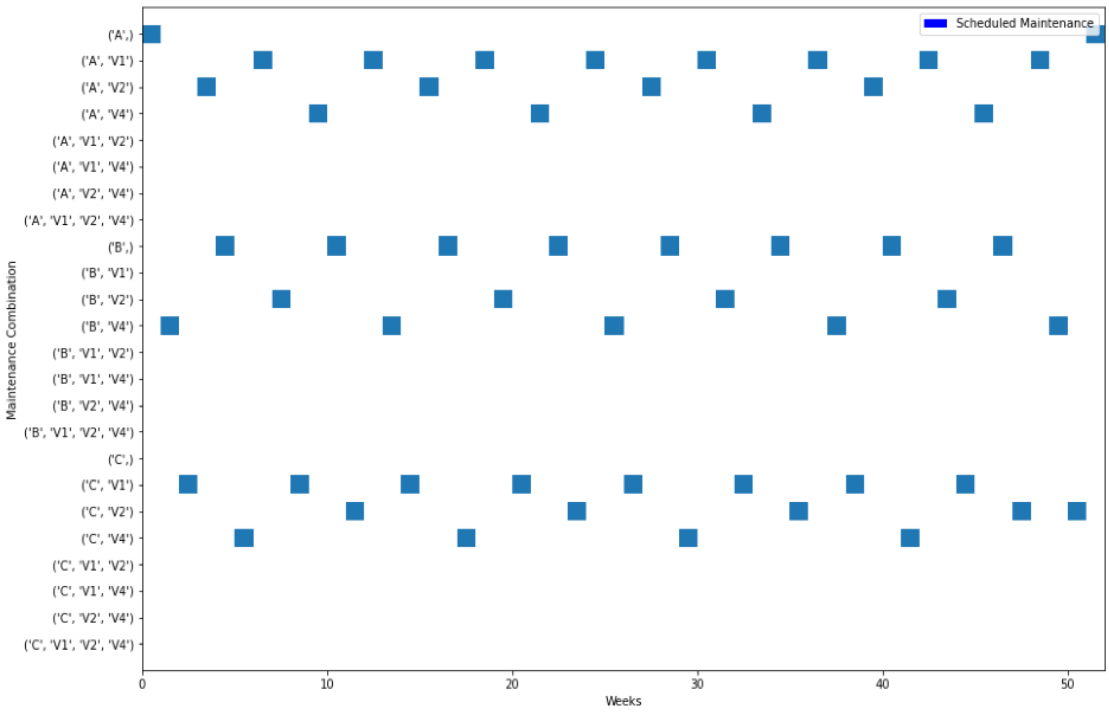

Figure 11: Optimal Long-Term Maintenance Schedule Excluding (B, V1) Short-Term Scenario

Table 11: Multi-Tier Framework Run-Times

	Situation 1	Situation 2
Multi-tier Phase	Time	Time
Initial Short-term Solves	2,911.83	2,921.52
Short-term Refinements	1,261.66	1,307.14
Long-term Solves	0.06	0.07
Total	4,173.55	4,228.73

Table 12: Multi-Tier Framework Objective Costs

	Initial Long-term Cost	Final Long-term Cost
Situation 1	$892,380.00	$892,920.00
Situation 2	$895,080.00	$895,620.00

CHAPTER III
SCHEDULING MAINTENANCE UNDER ADAPTIVE CIRCUMSTANCES

Abstract

The original work of (Deakins, et al. 2021) was based on the concept of closing the gap in the literature on the maintenance scheduling problem for nuclear power plant (NPP) systems using exact solution methods. These methods sought to minimize cost due to overtime while maintaining strict compliance with a plant's probabilistic risk assessment (PRA) program, and Technical Specifications (TS) Limiting Condition of Operation (LCO) timeframes.

This paper intends to build upon that original work by adding a real-world element of disruption to the weekly schedule. This process will include simulating a component failure at various times throughout the scheduled week, and requiring the program to respond accordingly. The original objective function to minimize cost of labor remains, along with the constraints to complete work within the week, however a new constraint is added upon failure of the random component, as the combination of out-of-service equipment yields a shorter LCO duration. Furthermore, more restrictive cut sets will exist in the PRA for this combination of unavailable equipment, requiring the program to generate and check a higher number of optimal or feasible solutions. Finally, a subsequent comparison between an industry scheduler and the programs results will be performed to analyze for potential cost savings, similar to the original paper.

Nomenclature

Parameters, Indices, and Sets

$i \in I$: All maintenance tasks for the time horizon.

$c \in C_R$: Components with maintenance tasks the system requires in the time horizon.

$c \in C_O$: Components with backlog maintenance tasks. (Common-train components).

$Q \in \varrho$: Sets of components that force the system to only one operable subsystem (train) if components share overlapping downtime.

$t \in T$: Hourly time steps: 1,...,T.

$j \in J$: Maintenance crew types.

$(i, i') \in P$: Maintenance tasks that have a precedence relationship, i.e., task i must precede task i'.

$G \in \mathcal{g}$: Sets of tasks that require the same man-hours, duration, crew type, and preceding and proceeding tasks.

$i \in B_C$: Tasks that bring a component c offline giving it a tag out status upon beginning.

$i \in E_C$: Tasks that remove the tag out status on component c upon completion.

$i \in F_C$: Tasks that designates a component c as operational upon completion.

$i \in A_j$: Maintenance tasks that require maintenance crew type j.

$R \in R$: All sets of minimal cut-sets for the system.

$s \in S$: Allowable shifts for crew members.

$d_t \in D$: The day to which time step t corresponds in the time horizon (MTWRF).

ES_i: Earliest possible start time of task i.

LS_i: Latest possible start time for task i.

H_i: Duration that task i requires for completion.

L_i: Number of crew members task i requires.

M_j: Total number of available crew members of crew type j.

P_{JS}: Weekly pay of a crew member of type j on shift s.

P_C: Projected future cost for completion of backlog maintenance for component c in a future week.

$|R|$: Size of set R.

$|Q|$: Size of set Q.

Variables:

$u_i(t)$: Assumes the value 1 if maintenance task i starts at time step t, 0 otherwise.

$v_i(t)$: Assumes the value 1 if maintenance task i is being performed during time step t, 0 otherwise.

$U_C(t)$: Assumes the value 1 if component c receives a tag out status at time step t, 0 otherwise.

$V_C(t)$: Assumes the value 1 if component c has a tag out status at time step t, 0 otherwise.

$W_C(t)$: Assumes the value 1 if component c has its tag out status removed at time step t, 0 otherwise.

X_C: Assumes the value 1 if backlogged component c is scheduled for maintenance, 0 otherwise.

$Y_C(t)$: Assumes the value 1 if a component c is offline at time step t, 0 otherwise.

$Z_C(t)$: Assumes the value 1 after if all maintenance for a component c is complete at time step t, 0 otherwise.

$\delta_i(t)$: Represents the number of crew members assigned to task i at time step t.

γ_{js}^{dt}: Represents the number of crew members of type j working an 8-hour shift s at time step t corresponding to day d.

Γ_{js}: Represents the total number of crew members of type j working a 5-day schedule on shift s.

λ_{js}^{dt} : Represents the number of crew members of type j working a 12-hour shift s at time step t corresponding to day d.

Λ_{js}: Represents the total number of crew members of type j working a 3-day schedule on shift s.

$\alpha(t)$: Assumes value 1 if the system has less two operable subsystems at time step t.

t_c^b: Represents the time step when a subsequent component breaks down.

Introduction

The original work of (Deakins, et al. 2021) was based on the concept of closing the gap in the literature on the maintenance scheduling problem for nuclear power plant (NPP) systems using exact solution methods. These methods sought to minimize cost due to overtime while maintaining strict compliance with a plant's probabilistic risk assessment (PRA) program, and Technical Specifications (TS) Limiting Condition of Operation (LCO) timeframes. This work utilized a two-stage SMILP whereby an optimal cost schedule is passed to a PRA model, in this case RAVEN, to analyze for cut set violations. Should a violation occur, this was passed back to the program as a new constraint, and the program calculated a new optimal schedule. This cycle was repeated until an optimal schedule was created with no PRA cut set violations. Lastly, the output of this program was checked against an actual industry scheduler, and analyzed for differences in technique, cost, and time to produce a solution. It was concluded that consistent cost savings of more than 20% could be attained through use of this model, while obtaining results in a fraction of the time required by the manual scheduling method.

This paper intends to build upon that original work by adding a real-world element of disruption to the weekly schedule. This process will include simulating a component failure at various times throughout the scheduled week, and requiring the program to respond accordingly. The original objective function to minimize cost of labor remains, along with the constraints to complete work within the week, however a new constraint is added upon failure of the random component, as the combination of out-of-service equipment yields a shorter LCO duration. Furthermore, more restrictive cut sets will exist in the PRA for this combination of unavailable equipment, requiring the program to generate and check a higher number of optimal or feasible solutions. Finally, a

subsequent comparison between an industry scheduler and the programs results will be performed to analyze for potential cost savings, similar to the original paper.

Motivation

This paper focuses on a multi-tier framework for a single week's schedule at a NPP that is disrupted by emergent work. To model the systems at a typical NPP, consider the following mock High Pressure Injection System (HPIS) shown in Figure 12 below. The HPIS consists of a water source, three identical injection pumps, discharge valves, and two identical injection lines leading to the Reactor Pressure Vessel (RPV). These components form three main subsystems or trains that are responsible for emergency water makeup to the RPV in the case of a transient where the reactor remains at or close to nominal operating pressure. This study does not consider RPV and RWST as any maintenance on these components requires the entire system to be brought offline.

The components can be arranged into subsystems or trains from a PRA perspective, such that a grouping of components required to establish an independent flow path can be readily identified and tied to one another's availability. To describe this HPIS system in these terms, components can be segregated into three trains: A, B, and C, with several common train components required for multiple or all subsystems to function. We consider components V1, V2, and V4 to be Common-train components since one may insert them into work windows involving two or more of trains above. They are also considered common as they are involved in the physical flow path for more than one discrete subsystem. Disruption of flow through one of these common components impacts the functionality of two or more subsystem. To this end, Table 13 below describes the categorization of this mock HPIS.

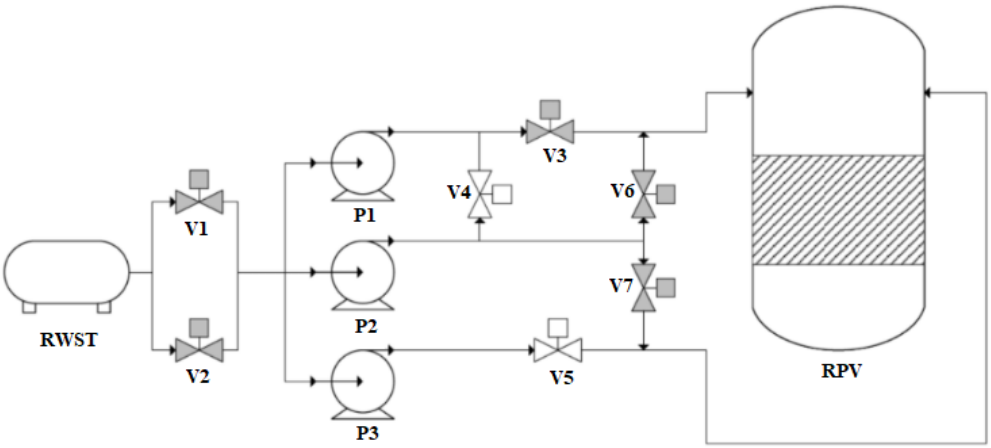

Figure 12: HPIS

Table 13: Categorization of Components in HPIS

A Train	B Train	C Train	Common
P1	P2	P3	RWST
V3	V6	V5	V1
	V7		V2
			V4

A common practice at NPPs is to schedule maintenance according to a corresponding division or train. This ensures that TS relationships between various systems and power supplies are preserved, and unnecessarily short duration LCOs and higher PRA categorizations are avoided. This does not however, preclude a scenario where, when one train is out of service for maintenance, a subsequent train experiences a fault requiring emergent corrective maintenance. Such scenarios are complicated, as the original LCO times must be met, however until one of the two trains can be restored, the NPP is under a more restrictive LCO, generally of 72 hours in duration or less. Once one system is restored, the plant is still under obligation to restore the remaining system within the original timeframe, regardless of the order in which subsystems are returned to service. This is known in the industry as Concurrent Timeclocks, and can be seen visually in Figure 13 below.

This condition provides a motivation for a robust, sophisticated scheduling methodology that can determine the shortest mechanism for restoration of one train of the affected safety system, while restoring both trains within the original timeframe. Furthermore, a program that can minimize cost will benefit the utility, as these emergent maintenance situations are expensive propositions. This framework continues the original work of (Deakins, et al. 2021), by applying the original optimization model where O&M cost is minimized, with added constraints for performing the emergent work, and restricting the combined out of service time to the new LCO timeframe.

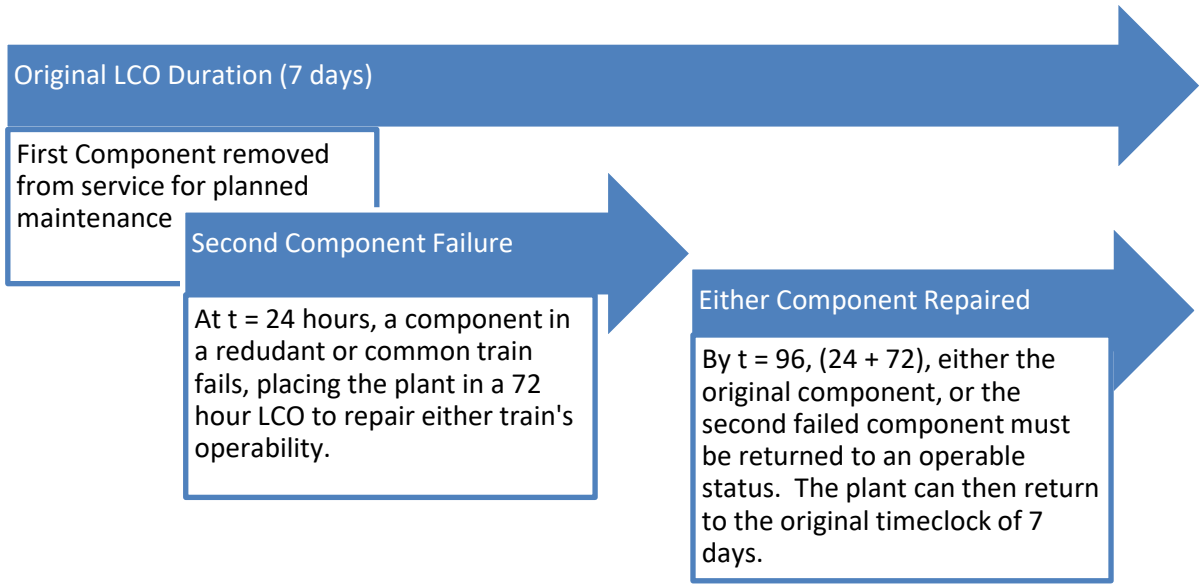

Figure 13: Concurrent Timeclock Methodology

Methods

Framework

The framework produces an optimal cost maintenance schedule and identifies reliability concerns with each new incumbent solution. The two-tier framework contains an integer programming (IP) model that is responsible for producing an optimal cost maintenance schedule and a PRA model to analyze incumbent scheduling solutions for reliability cut-set violations. A reliability cut set is a set of components that if there is an overlap in their respective downtimes, then the reliability of the system falls below a designated threshold. Therefore, a violation is any schedule where all components of any cut-set have overlapping downtime. When and if cut set violations occur, cutting planes are generated to remove these violations from the next incumbent solution. Figure 14 below illustrates the core integration of the IP and the RAVEN PRA models into a capable scheduling program.

Input data describing the system and maintenance tasks associated with the system is provided into the framework. The program then generates an IP model instance and a PRA model instance. The framework solves the IP model by way of branch and cut. A maintenance schedule is then generated for each new incumbent solution, which is passed to RAVEN to evaluate against the PRA model. RAVEN will identify whether any cut-set violations exist, and these will be passed back to the main framework for cut-set constraint generation. Once all necessary cut-set constraints are generated, the IP solver accepts the constraints by means of a callback utility. Each cut-set constraint is added before subsequent branch and cut operations. The two-tier framework concludes in an optimal cost maintenance schedule with no cut-set violations.

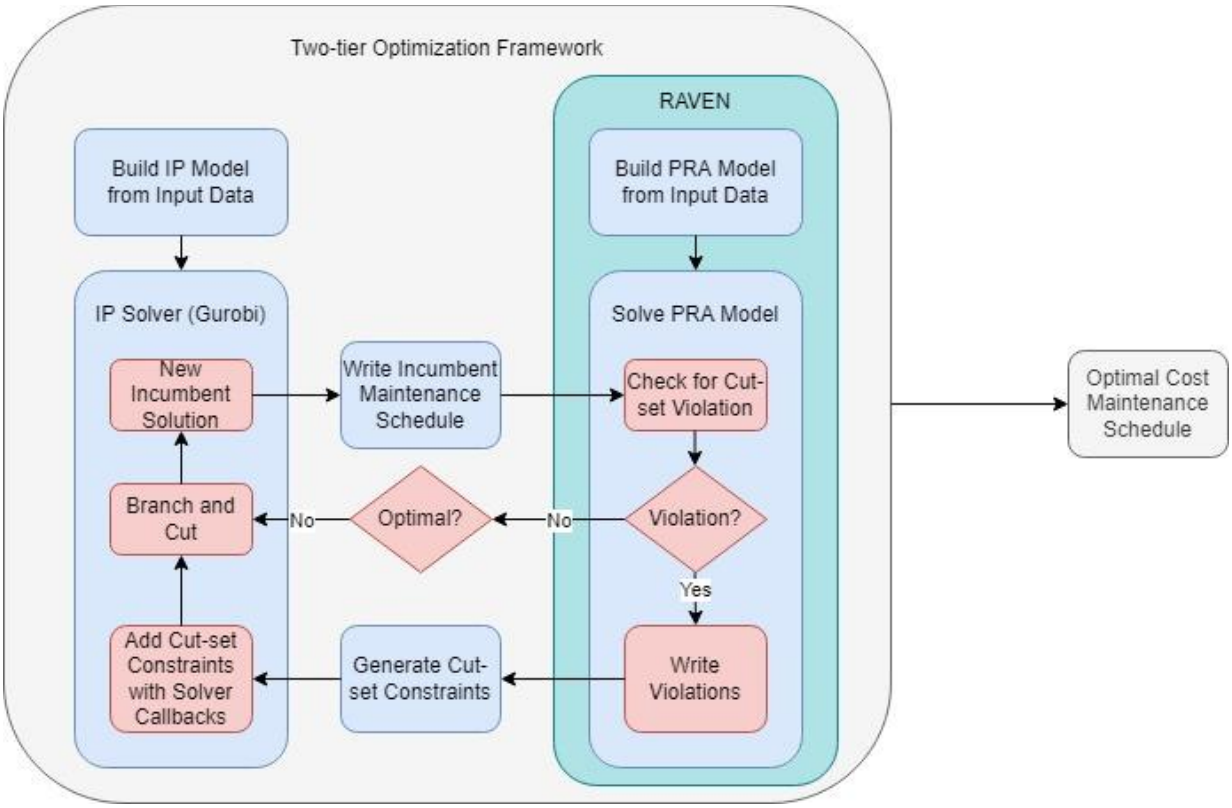

Figure 14: Two-Tier Optimization Framework

Model

Minimize:
$$\sum_{j \in J} \sum_{s \in S} P_{js}(\Gamma_{js} + \Lambda_{js}) \quad (1)$$

Subject to:
$$\sum_{t = ES_i}^{LS_i} u_i(t) = 1 \qquad \forall i \in I(c), \forall c \in C_R \quad (2)$$
$$\sum_{t = ES_i}^{LS_i} u_i(t) = X_c \qquad \forall i \in I(c), \forall c \in C_O \quad (3)$$
$$\sum_{t\prime = t - H_i + 1}^{t} u_i(t') = v_i(t) \qquad \forall t \in T, \forall i \in I \quad (4)$$
$$L_i v_i(t) = \delta_i(t) \qquad \forall t \in T, \forall i \in I \quad (5)$$
$$\sum_{t \in T} \delta_i(t) \leq \gamma_{js}^{dt} + \lambda_{js}^{dt} \qquad \forall j \in J, \forall i \in A_j, \forall s \in S \quad (6)$$
$$\gamma_{js}^{dt} \leq \Gamma_{js} \qquad \forall j \in J, \forall s \in S, \forall d_t \in D \quad (7)$$
$$\sum_{dt \in D} \gamma_{js}^{dt} = 5\,\Gamma_{js} \qquad \forall j \in J, \forall s \in S \quad (8)$$
$$\lambda_{js}^{dt} \leq \Lambda_{js} \qquad \forall j \in J, \forall s \in S, \forall d_t \in D \quad (9)$$
$$\sum_{dt \in D} \lambda_{js}^{dt} = 3\,\Lambda_{js} \qquad \forall j \in J, \forall s \in S \quad (10)$$
$$\sum_{s \in S} \Gamma_{js} = \Lambda_{js} \leq M_j \qquad \forall j \in J \quad (11)$$
$$\sum_{t\prime = t}^{LS_i} u_i(t') + \sum_{t\prime = ES_{i\prime}}^{t + H_i - 1} u_{i\prime}(t') \leq 1 \qquad \forall (i, i') \in P, \forall t \in T \quad (12)$$
$$u_i(t) = U_c(t) \qquad \forall t \in T, \forall i \in B_c, \forall c \in C_R \cup C_O \quad (13)$$
$$u_i(t - H_i) = W_c(t) \qquad \forall t \in T, \forall i \in E_c, \forall c \in C_R \cup C_O \quad (14)$$
$$u_i(t - H_i) = Z_c(t) \qquad \forall t \in T, \forall i \in F_c, \forall c \in C_R \cup C_O \quad (15)$$
$$U_c(t) - W_c(t) = V_c(t) - V_c(t - 1) \qquad \forall t \in T, \forall c \in C_R \cup C_O \quad (16)$$
$$U_c(t) - Z_c(t) = Y_c(t) - Y_c(t - 1) \qquad \forall t \in T, \forall c \in C_R \cup C_O \quad (17)$$
$$\sum_{t = ES_i}^{LS_i} v_i(t) = D_i \qquad \forall i \in I(c), \forall c \in C_R \quad (18)$$

$$\sum_{t=ES_i}^{ES_i} v_i(t) = D_i X_c \qquad \forall i \in I(c), \forall c \in C_O \qquad (19)$$

$$\sum_{t=ES_i}^{LS_i} t \cdot u_i(t) \leq \sum_{t=ES_{i'}}^{LS_{i'}} t' \cdot u_{i'}(t) \qquad \forall (i, i') \in G \qquad (20)$$

$$\sum_{c \in C_R} \sum_{i \in B_c} u_i(0) = 1 \qquad (21)$$

$$\sum_{c \in Q} V_c(t) \leq (|Q| - 1) + \alpha(t) \qquad \forall t \in T, \forall Q \in q \qquad (22)$$

$$\sum_{t'=t}^{t+72} \alpha(t') \leq 72 \qquad \forall t \in T \qquad (23)$$

$$\sum_{c \in R} V_c(t) \leq |R| - 1 \qquad \forall t \in T, \forall r \in R \qquad (24)$$

$$u_i(t), v_i(t), U_c(t), V_c(t), W_c(t), X_c, \alpha(t) \in \{0, 1\} \qquad (25)$$

$$\gamma_{js}^{dt}, \lambda_{js}^{dt}, \Gamma_{js}, \Lambda_{js} \in \{0, 1, 2, \ldots, \infty\} \qquad (26)$$

$$\sum_{t'=t_b^c}^{tb+72} Z_c(t') = 1 \qquad \forall c \in C^b \qquad (27)$$

The objective in (1) minimizes the cost of labor and is a departure from the original model where the projected future cost of backlog maintenance is no longer credited. This is because the back log maintenance is now replaced with emergent maintenance, which must be performed within the required timeframe of 72 hours. Constraint (2) ensure that each maintenance task the system requires begins in the time interval [ES_i, LS_i]. Constraint (3) ensures that if a backlog maintenance component *c* appears in the maintenance schedule, then all maintenance tasks *i* for that component begin in the time interval [ES_i, LS_i]. Constraint (4) ensures that a task completes in consecutive time steps. Constraint (5) requires that the correct number of maintenance crew members to complete any task found in the scheduling solution. Constraint (6) restricts the number of crew members the model assigns at any time step t to no more than the number of crew members

the model assigns to the day and shift corresponding to that time step. Constraint (7) restricts the number of crew members the model assigns to an 8-hour shift *s* on any day d_t to no more than the total number of crew members the model assigns to a 5-day week on shift s. Constraint (8) ensures each crew member the model assigns to an 8-hour shift *s* works five of those shifts in the week. Constraints (9) and (10) perform the same function of the previous two constraints except for the crew members that the model assigns to a 3-day work week of 12-hour shifts. Constraint (11) ensures the total number of crew members the model assigns to five 8-hour shifts and three 12-hour shifts is no more than the available number of crew members within the NPP for each crew type.

Constraint (12) restricts the starting time step of a task *i'* to be after task *i* is complete when these two tasks have a precedence relationship $i \prec i'$. Each component the model schedules for maintenance has both a task that brings that component offline and a task that brings that component online. In industry this is referred to as placing and removing a tagout or lockout on these components. Constraint (13) ensures that a component is brought offline and receives a tagout at the same time step *t* that its tagout task occurs. Constraint (14) removes a component's tagout at the same time step that its tagout removal task is complete. Constraint (15) designates a component to be available from a PRA perspective when the final maintenance task for that component is complete, including the tagout removal. Constraint (16) ensures that a component retains its tagout until the task that removes the tagout status is complete. Constraint (17) ensures a component remains offline until the final maintenance task for that component is complete. Constraints (18) and (19) help to tighten the model for all maintenance tasks the model schedules. Constraint (20) helps to break the symmetry of scheduling maintenance tasks that require the same

man-hours, duration, crew-type, and preceding/proceeding tasks. Constraint (21) ensures that one of the components that has maintenance tasks the system requires is brought offline at time step 1. Constraint (22) forces the value of $\alpha(t)$ if the maintenance schedule moves the system into a 72-hour LCO window. Specifically concerning the system in this paper, $\alpha(t)$ assumes the value 1 when only one of the three system trains is operable per the simulated system licensing basis. In this case, the system can remain in this state for no longer than 72 hours (time steps) and the model enforces this with Constraint (23). We list Constraint (24) in the formal description of the model; however it is actually a cutting plane constraint that the two-tier framework adds to the model at each new incumbent solution if a cut-set violation occurs in that solution.

Implementation

The main programming language for the two-tier framework is Python. Referring to Figure 14, the outermost block can be thought of as a python driver program that constructs the IP model with the PYOMO modeling language, writes any data necessary for the program, and calls RAVEN to perform PRA analysis. The Python programming language presents itself as the most advantageous to this study as PYOMO and RAVEN both have a Python foundation. The following paragraphs break down the two-tier framework to its individual blocks and describe the process for building that specific piece of the program.

First, we construct the IP model section of Figure 14 with the PYOMO modeling language. PYOMO is a complete python implementation of (Deakins, et al. 2021)a powerful optimization modeling language that can be used with any open-source or commercial solver that has the capability to read an “lp” or “mps” file. For more information of the PYOMO modeling language, see (Hart, et al. 2017). We create functions in the driver program to construct the PYOMO model

that is the result input data that describes maintenance tasks and the system of interest. All input data are read from a spread sheet file. A complete description of the input data is contained in the Tests and Results section of this paper. The framework constructs the model using Python functions and PYOMO modeling language techniques and passes it to Gurobi, a commercial optimization solver.

Next, the main driver code controls the RAVEN portion of Figure 3.3. In this study, we use a RAVEN built-in cut set solver to perform the PRA analysis. We construct the driver in such a way that it easily communicates scheduling solutions with RAVEN. RAVEN is a powerful reliability analysis tool and is capable of reading data from "csv" files. With this, the framework generates csv files at each new incumbent IP solution that encode the scheduling information and passes this csv file as input to RAVEN.

Finally, the driver code integrates the PYOMO model and the RAVEN model by writing schedules from the IP solution and reading cut-set information from RAVEN. The two-tier framework calls RAVEN as a sub-process at each new incumbent solution by executing a callback function within Gurobi. Most modern commercial solvers allow users to define callback functions that evaluate and perform certain function given the status of the solver. Once a new incumbent is found, Gurobi uses the callback function to access the internal solver values for the variables and writes them as a maintenance schedule to an exterior csv file. The same callback function then invokes RAVEN to read that csv file and analyze for cut-set violations. RAVEN also requires an "xml" file that it interprets to construct the cut-set solver model and identify to report its findings. Once RAVEN reports its findings, the original callback function parses the RAVEN output and uses any violated cut-sets to generate constraints and then add them to the model. Once

complete, Gurobi continues the branch-and-cut process solving the IP model with the additional cut-set constraints.

A major departure from the original work here is that at any time step during the week, a subsequent component will be removed from service due to an identified failure mechanism. These failures can be detected as part of normal rounds performed on the equipment performed by the Operations personnel at the NPP, or resulting from regular periodic surveillance testing to maintain the equipment in an operable status. Typically, these surveillances will not be scheduled with redundant equipment removed from service for maintenance in an effort to deter this exact scenario. However, certain equipment checks must be performed on a daily basis and cannot be avoided. Referring to the HPIS model in Figure 12, Valves 1, 2, and 4 were selected for testing purposes at time steps of t20, t60, and t90. Valves 1 and 2 will place the NPP in a 72-hour LCO scenario, invoking Constraint (23) and forcing the program to solve a new schedule in which the overlapping maintenance windows are restricted to 72-hours, while completing all original work within the original week. The major difference here is that instead of planning from t0 to restrict the work window to something tighter than the 5-day work week, the program must reanalyze at t20, t60, and t90 to restore at least one subsystem to operable status.

Tests and Results

Utilizing the original work in (Deakins, et al. 2021), an already optimized schedule for the A Train of HPIS was uploaded to the framework. This was used as a baseline starting point, assuming the NPP would enter the given work week with a plan to execute this optimized schedule. At times t=20, t=60, and t=90, failures of V1, V2, and V4 were inputted as new required work. These specific failures were chosen as they are more likely types of failures to be discovered during

the conduct of the maintenance on P1 and V3. V1 and V2 are shown as normally closed valves that would receive an open signal on system initiation. Therefore, leak-by past V1 and V2 would be detected through normal Operator rounds by level lowering in the RWST. Furthermore, V4 would be closed for any maintenance done on P1 and V3 to preclude introducing a flow path into the work boundary should a system initiation be received. Leakage into this boundary would be detected by the work group, and V4 would be declared inoperable, requiring emergent work. Failures of P2 and P3 are unlikely in this scenario, as they are in a standby condition. Such failures would also unlikely go detected until a system initiation signal is received, and therefore were not considered in the following simulations. These failures could be inserted and solved for as well due to the normal precedence relationships that exist between tasks as described in the model.

Figure 15 shows a schedule that was optimized for A Train maintenance until t=20, at which point V1 is declared inoperable and requires emergent work. For the purposes of this demonstration, the same work scope that was used in Deakins original work was inserted for work on V1. This is to drive a level of consistency and allow for fair comparisons to be made between the emergent and non-emergent scenarios. The failure type would be unknown, however the particular tasks are broad in scope and would apply to a wide range of failures. These include the component tagout steps, valve disassembly and inspection, a repair task dubbed "Limitorque Rebuild" in this case simulating maintenance on the valve actuator, valve reassembly and stroking, and tagout removal and testing. It can be seen here that the program constrains the window to 72 hours, and does not provide any conflict in workgroup overlap ie, the same workers are not double booked with tasks presenting a non-feasible solution. Figures 16 and 17 below demonstrate a similar condition for failures of V1 at t=60 and t=90 respectively.

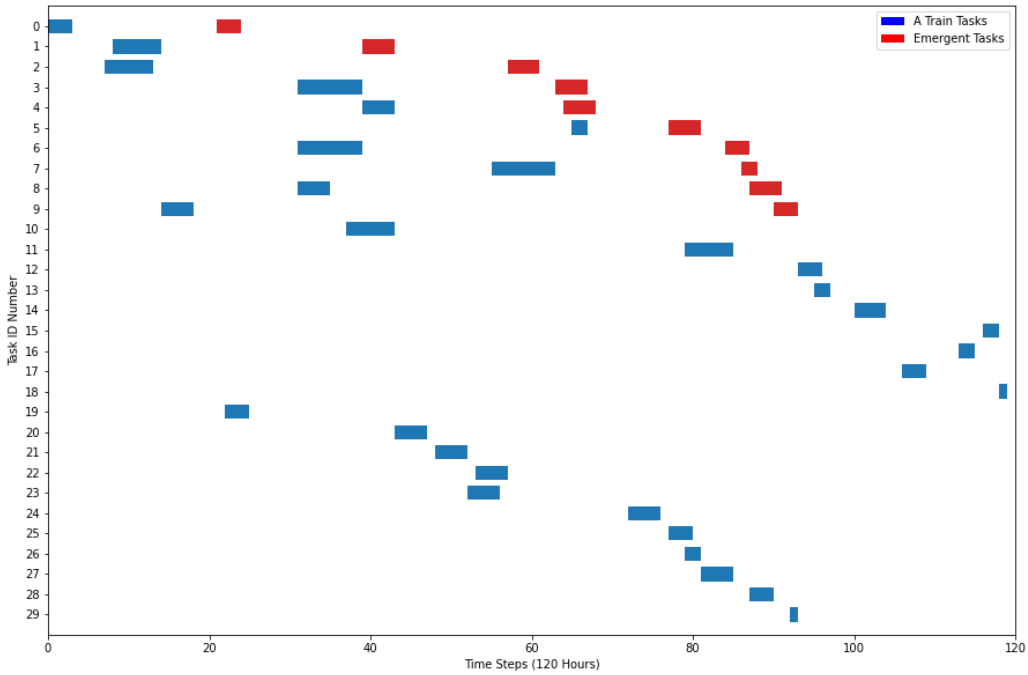

Figure 15: V1 Failure at t=20 During A-Train Window

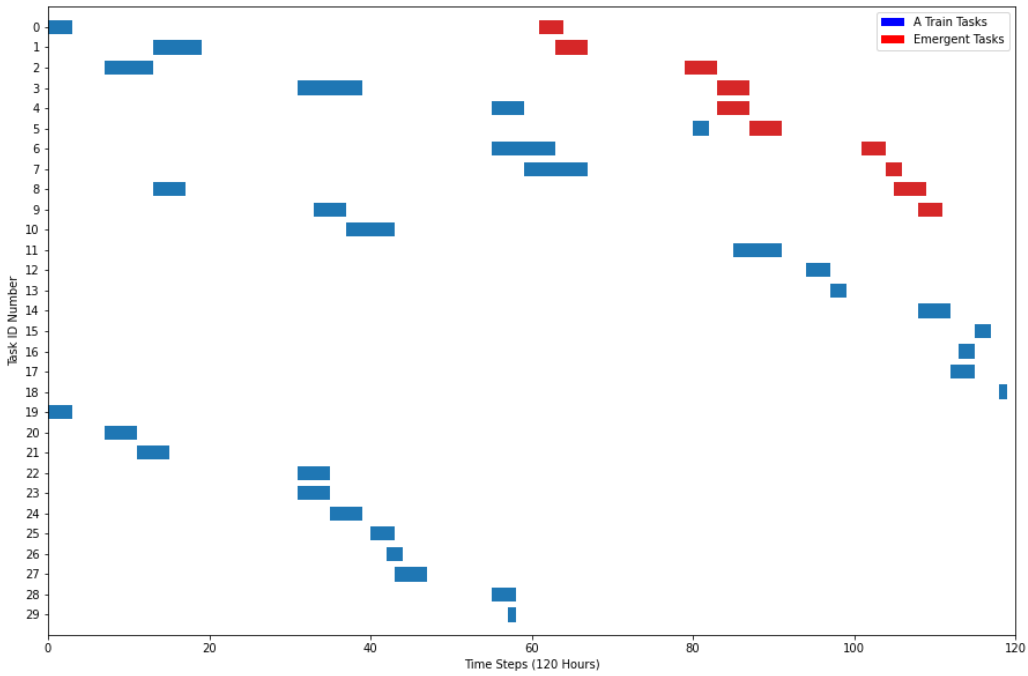

Figure 16: V1 Failure at t=60 During A-Train Window

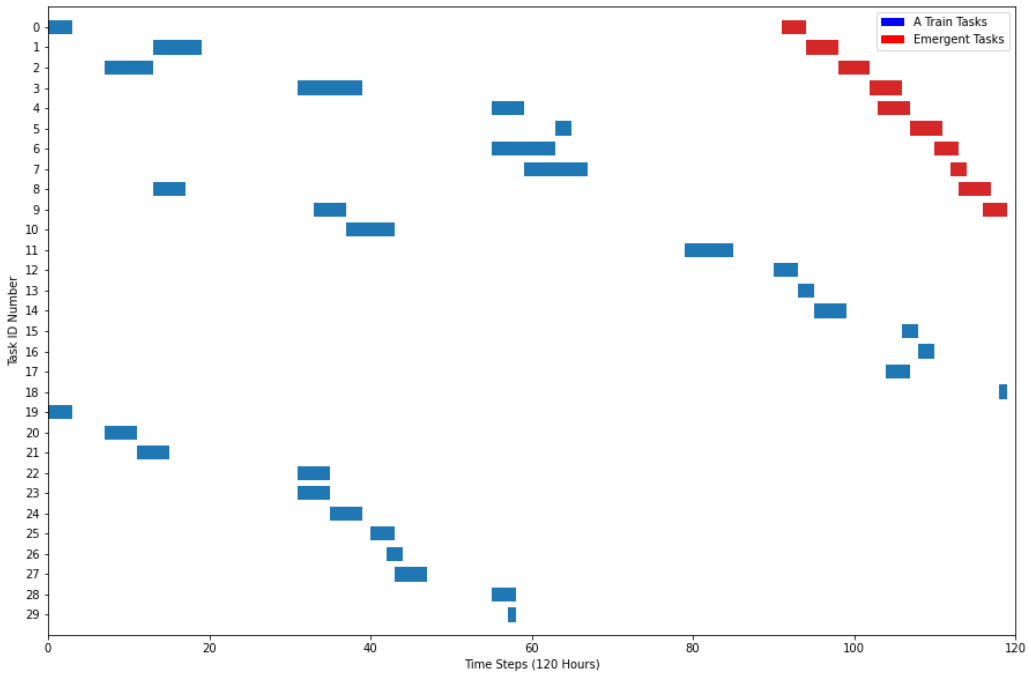

Figure 17: V1 Failure at t=90 During A-Train Window

From Deakins original work, we know that an optimized A Train schedule resulted in an objective function cost of $28,920. This cost invariably will increase with the introduction of emergent work, however the objective function is primarily dependent in this case on what time step the failure occurs. This is due to Constraints (6), (7), and (8), which ultimately constrain the problem to solve within the normal 5-day work week. Future work here will consider the possibility of extending the work week to include the weekend, at the penalty of paying industry standard premiums of 1.5X and 2.0X pay for Saturday and Sunday work respectively. The objective functions for the three scenarios above when compared to the original schedule are shown below in Table 14. Also depicted in Table 14 are the times to the various solutions of our problem. It can be seen that the later a failure occurs in the given time parameters, the faster a solution is arrived at by the program. This makes sense as there are physically fewer potential solutions to test, yielding an overall reduction in solution times. This finding will also be up for consideration in future work where the weekend can be utilized at a penalty.

Finally, for a complete continuation of Deakins original work, a comparison between currently employed industry techniques for scheduling and the results of this program is performed. These results are shown below in Figures 18, 19, and 20. For these scenarios, two basic solutions would be considered, where both involve staffing all effected groups around the clock. The first is the amount of time and cost associated with rapidly completing the original work window, and then moving on to the emergent work. The second scenario would be to divert all applicable resources to restoring the failed component, and then going back to completing the originally scheduled work.

Table 14: Objective Function Results for V1 Failure Scenarios

Scenario	Original	Failure at t=20	Failure at t=60	Failure at t=90
Obj Func Cost	$28,920	$30,840	$31,800	$33,480
Time to Solution	N/A	535	62	6

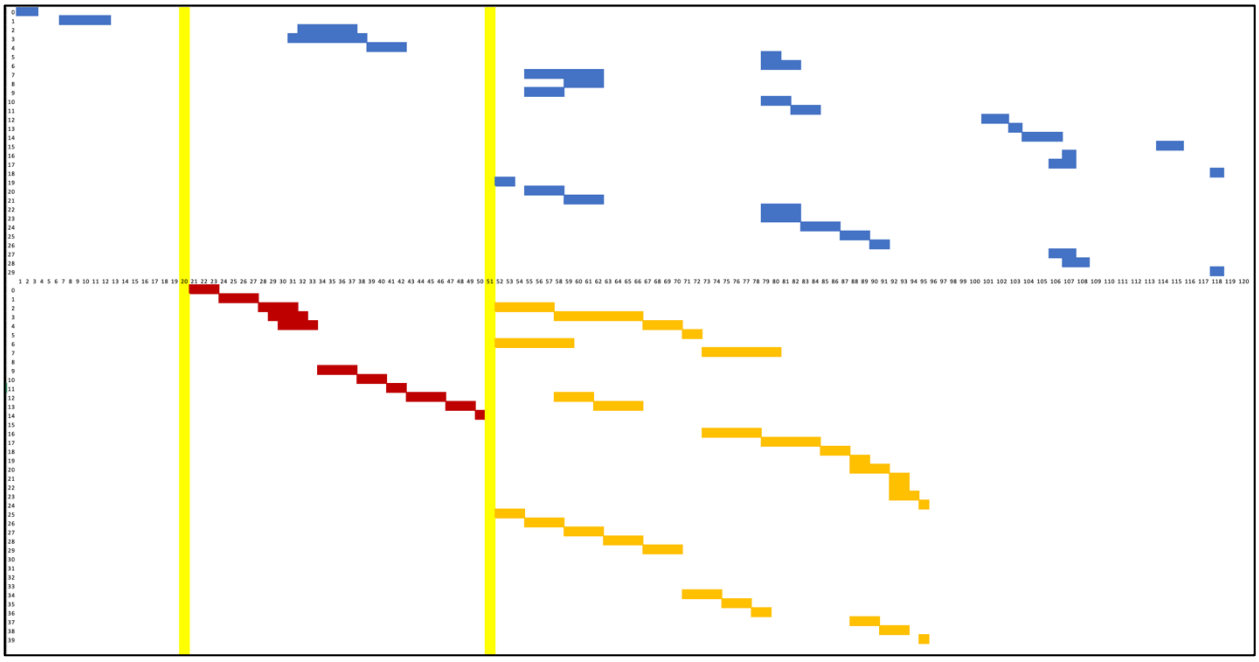

Figure 18: Hand Drawn Schedule Adjustment for V1 Failure at t=20

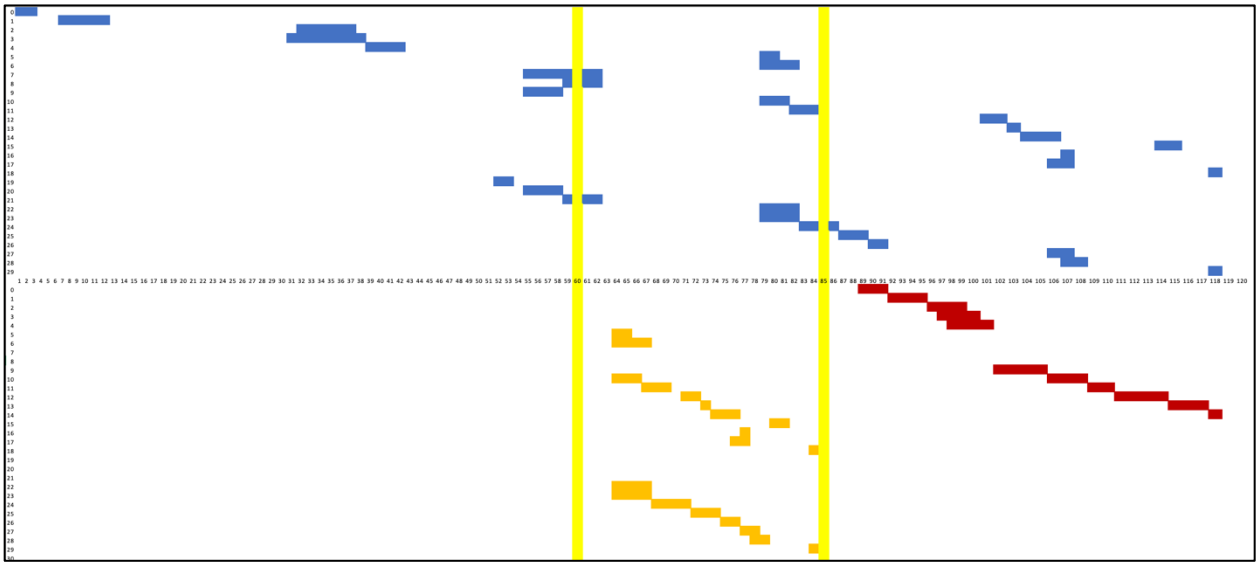

Figure 19: Hand Drawn Schedule Adjustment for V1 Failure at t=60

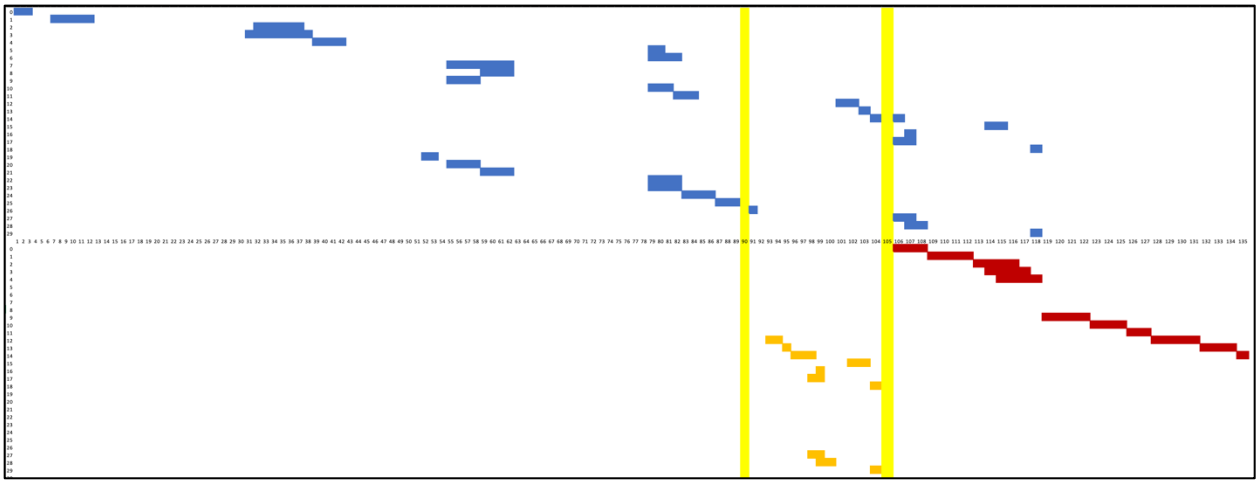

Figure 20: Hand Drawn Schedule Adjustment for V1 Failure at t=90

As shown in the three figures above, options for manual scheduling relegated a hold in the schedule for P1 and V3 at t=20 in order to rapidly restore V1 to service. Contrast this with failures at t=60 and t=90, where it was deemed prudent to complete the work on P1 and V3 first, exiting the 72-hour LCO, and then redirecting workers to restore V1. This strategy was selected due to the remaining work on the original schedule being able to be completed in a shorter timeframe in both latter cases than the overall window for a V1 OOS window. In the case of t=20, very little work had been performed in the original window, and therefore a shift to V1 at that time allowed exiting the LCO window in a shorter overall time. One interesting detail of note, the manual scheduling method did not produce results where both OOS windows could be completed by t=120, and required expanded timeframes to t=135. This represents work on Saturday of the given work week, and would be paid at a premium. Results in cost differential are shown below in Table 15, including the effects of weekend work.

Discussion

This study builds on Deakins original work, where a two-tier framework for modeling and solving maintenance scheduling problems in NPPs was presented and adds the uncertainty inherent to these schedules by simulating the failure of additional system components. We construct an integrated approach to handle an IP optimization model that couples with a PRA model to ensure optimal cost, reliable maintenance schedule with adaptability to changing plant conditions. This two-tier framework allows RAVEN to identify cut-set violations as they occur, and also allows the IP model to add constraints to remove these cut-set violations via a callback function. We construct data sets for this study by mapping real-work NPP maintenance data onto a hypothetical

HPIS that might be found in a NPP. We collect this real-world data from previous maintenance workload data sets in the NPP through which our point-of-contact is employed.

The framework is validated by solving the maintenance scheduling problem for the major trains of the HPIS, utilizing three different failed components at three different times during the simulated work week. Additionally, a comparison is made between the optimal schedule returned by the program and a hand-generated schedule in accordance with the current maintenance scheduling techniques used by out point-of-contact in the industry. We conclude that this methodology would serve as a benefit to schedulers and decision makers at both the NPPs themselves, as well as at corporate services groups for validation and challenge purposes as part of the nuclear oversight function. We also conclude that there is reason to continue further research in automated methods of maintenance scheduling within NPPs at it pertains to opening up the schedule to potential weekend work, or longer term refuel outage (RFO) scheduling. Furthermore, for a true assessment of the potential cost-savings that could be achieved, the framework should incorporate a complete maintenance workload for all systems undergoing maintenance in a given week. This paper shows promising early results for this framework and provides a foundation for further research into using exact methods for optimal maintenance scheduling in NPPs.

Table 15: Objective Function Comparison

Scenario	t=20 Failure	t=60 Failure	t=90 Failure
Obj Function Cost	34,048.00	40,166.67	39,301.54
% Difference	10.40%	26.31%	17.39%
Savings	$3,208	$8,366.67	$5,821.54

APPENDIX

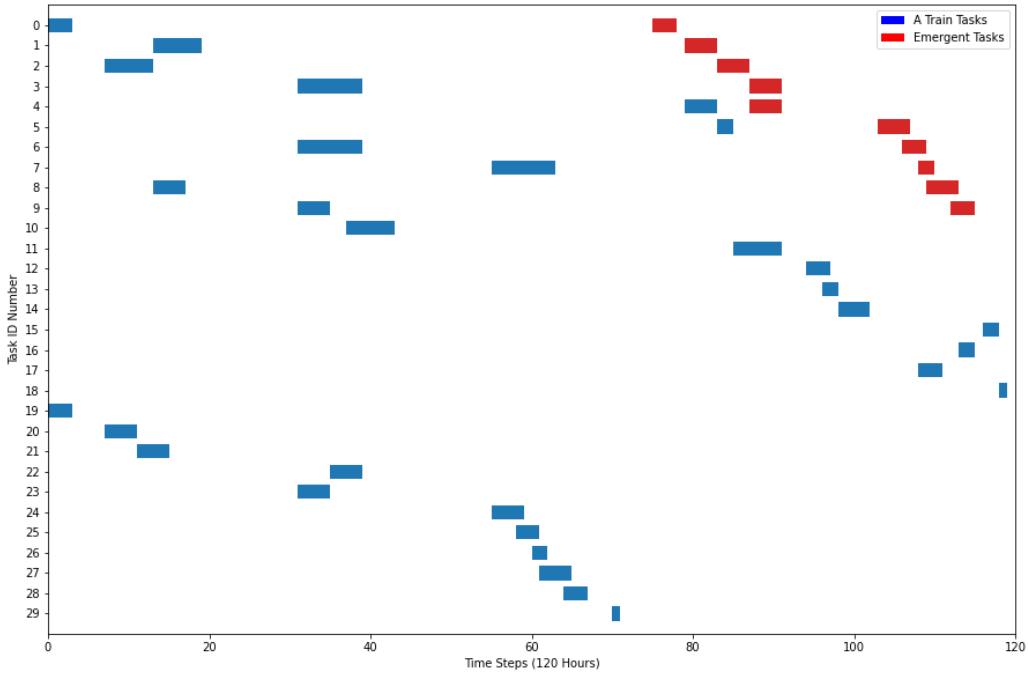

Figure 21: V2 Failure at t=20 During A-Train Window

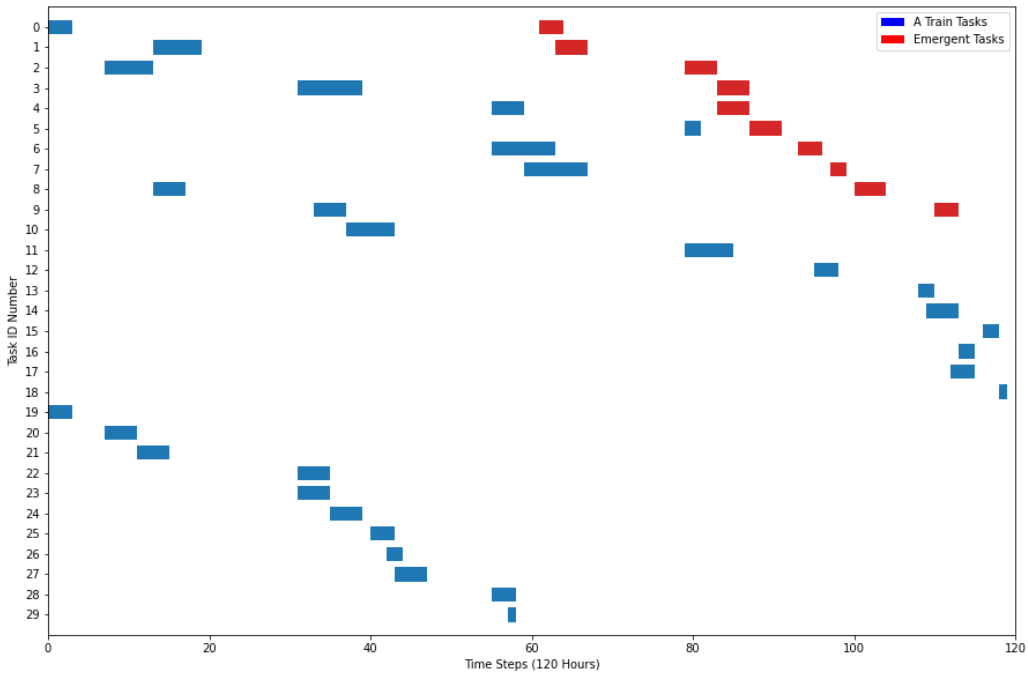

Figure 22: V2 Failure at t=60 During A-Train Window

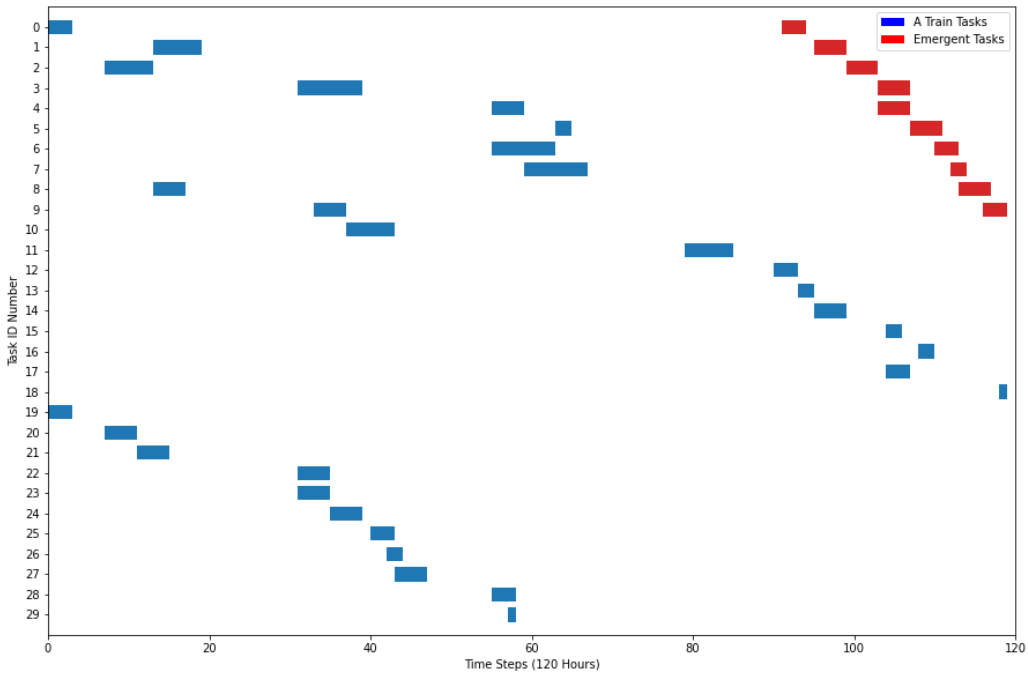

Figure 23: V2 Failure at t=90 During A-Train Window

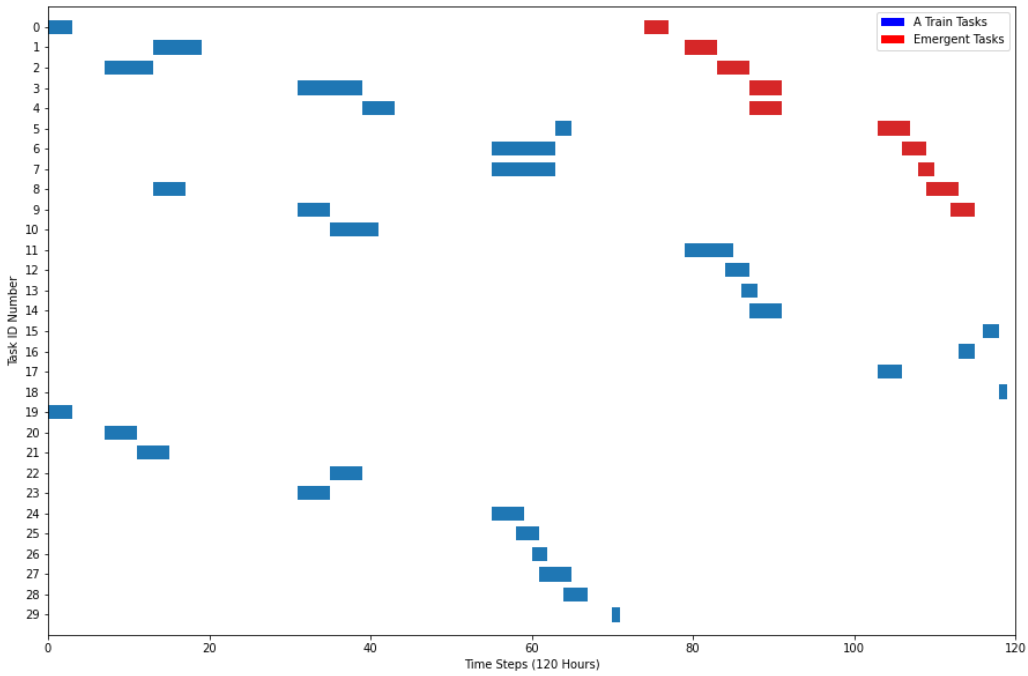

Figure 24: V4 Failure at t=20 During A-Train Window

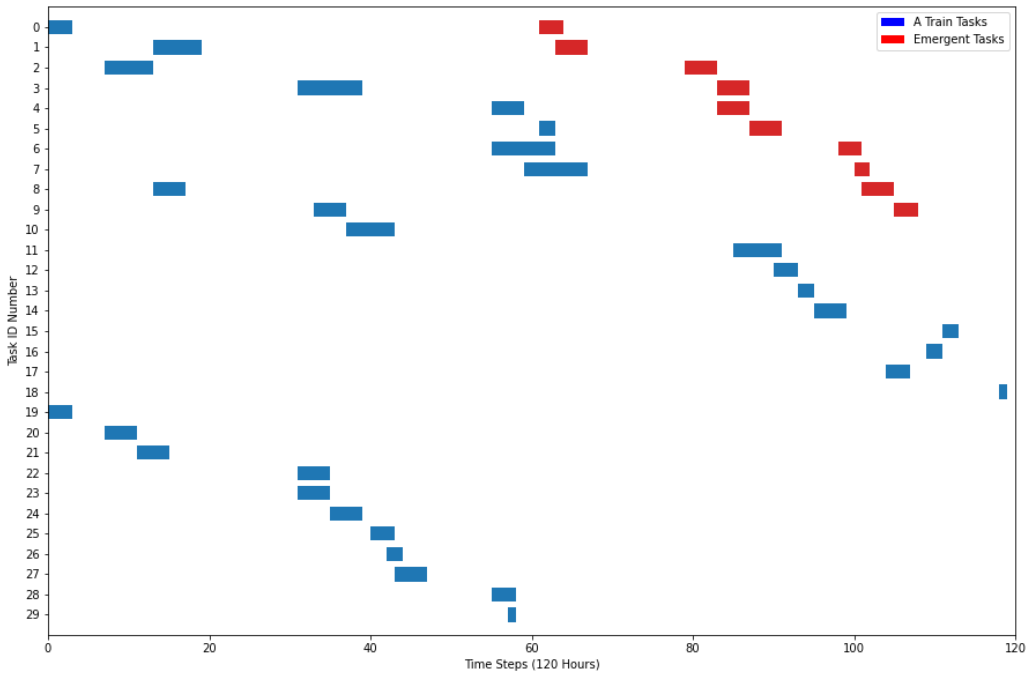

Figure 25: V4 Failure at t=60 During A-Train Window

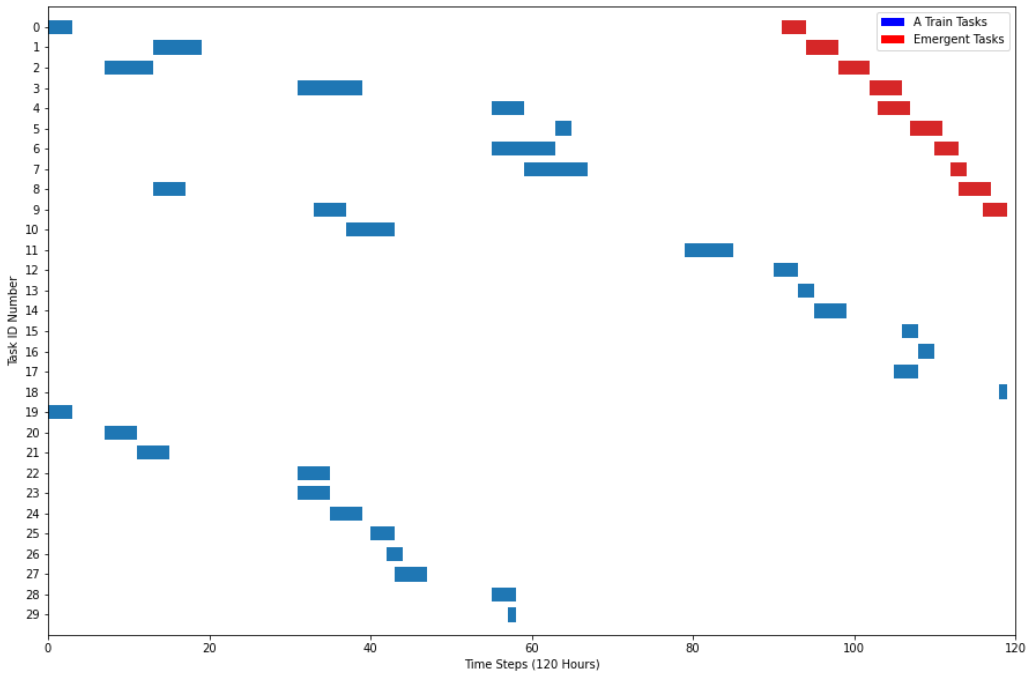

Figure 26: V4 Failure at t=90 During A Train Window

CONCLUSION

The research contained herein developed a 3-prong approach to optimizing maintenance schedules in modern Nuclear Power Plants. The paper grows and develops on a basic weekly schedule who's objective function is to minimize the expenditure of the salary component of O&M costs. This is performed by prioritizing work that does not incur additional or penalty costs due to working off-normal hours on a model of a NPP, while adhering to a set of constraints modeled after regulatory requirements for out-of-service times for nuclear systems. It was shown that a cost savings as much as 21% can be realized through such a system of scheduling maintenance.

The research goes on to explore longer time horizons than a simple weekly schedule. The importance of this is in showing that an entire system, not only the individual subsystems, can have an optimized maintenance schedule that maintains compliance with preventive maintenance due dates. The concept of grace period is also explored here, leaving room for decision making within the model as it seeks optimality. Here it was shown that an objective function of 892K on a goal of < $1.5M was achieved, which was the 52-week extrapolation cost of the original weekly cost from the first segment of this research.

Finally, the third chapter of this work demonstrated the ability of the program to respond to emergent maintenance needs. This represents the capacity to schedule more physical tasks within a shorter timeframe, still achieve optimality, and provide better results than the current industry practices. A cost savings of 17% - 26% were achieved based on a series of potential system failures when compared to the solutions achieved from the industry scheduler. Therefore, in all cases not only was optimality achieved, it was shown by physical comparison that there are advances to gained in the nuclear industry by implementation of the methods studied here.

BIBLIOGRAPHY

Aghaie, M., A. Norouzi, A. Zolfaghari, A. Minuchehr, Z. Mohamadi Fard, and R. Tumari. 2013. "Advanced progressive real coded genetic algorithm for nu- clear system availability optimization through preventive maintenance scheduling." *Annals of Nuclear Energy* 60: 64 - 72.

Alfonsi, A., C. Wang, P. Talbot, M. M. Abdo, M. Gamal, D. Mandelli, C. Rabiti, J. J. Cogliati, and R. A. Kinoshita. 2021. "Raven User Manual." Idaho Falls,, ID: Idaho National Lab (INL).

Ayoobian, N, and M. Mohsendokht. 2016. "Multi-objective optimization of maintenance programs in nuclear power plants using genetic algorithm and sensitivity index decision making." *Annals of Nuclear Energy* 88: 95 - 99.

Berrichi, A., F. Yalaoui, L. Amodeo, and M. Mezghiche. 2010. "Bi- objective ant colony optimization approach to optimize production and maintenance scheduling." *Computers & Operations Research* 37(9): 1584 – 1596.

Canto, S. P. 2008. "Application of benders' decomposition to power plant preventive maintenance scheduling." *European journal of operational research* 184: 759 - 777.

Carlos, S., A. Sanchez, S. Martorell, and J-F Villanueva. 2012. "Particle swarm optimization of safety components and systems of nuclear power plants under uncertain maintenance planning." *Advances in Engineering Software* 50: 12 - 18.

Chou, Q., D. Ge, and R. Zhang. 2014. "Pso based optimization of testing and maintenance cost in NPPs." *Science and Technology of Nuclear Installations.*

Coble, J, P Ramuhalli, L J Bond, J Hines, and B Ipadhyaya. 2015. "A review of prognostics and health management applications in nuclear power plants." *International Journal of prognostics and health management,* 6:016.

Deakins, E., N. Akhundov, J. Coble, T Gallacher, A. Khojandi, D. Mandelli, and J. Ostrowski. 2021. "An Exact Method for Maintenance Schedule Optimization in NPPs."

Decker, R. 1996. "Applications of maintenance optimization models: a review and analysis." *Reliability engineering & system safety* 51(3): 229–240.

Dopazo, J., and H. Merrill. 1975. "Optimal generator maintenance scheduling using integer programming." *IEEE Transactions on Power Apparatus and Systems* 94(5): 1537–1545.

Froger, A., M. Gendreau, J. E. Mendoza, E. Pinson, and L. M. Rousseau. 2016. "Maintenance scheduling in the electricity industry: A literature review." *European Journal of Operational Research* 251(3): 695–706.

Ge, D., S. Chen, Z. Wang, and Y. Yang. 2018. "Particle swarm evolutionary computation-based framework for optimizing the risk and cost of low demand systems of nuclear power plants." *Journal of Nuclear Science and Technology* 55: 19 - 28.

Guiberteau, P. 2020. "Chapter I." *GIF 2020 Annual Report.*

Hadavi, S. M. H. 2008. "Risk-based, genetic algorithm approach to optimize outage maintenance schedule." *Annals of Nuclear Energy* 35: 601 - 609.

Hart, W. E., C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola. 2017. *Pyomo-optimization modeling in python.*

Harunuzzaman, M., and T. Aldemir. 1996. "Optimization of standby safety system maintenance schedules in nuclear power plants." *Nuclear Technology* 113(3): 354 - 367.

IAEA. 2021. *IAEA Bulletin.*

IAEA. 2020. "Advances in small modular reactor technology developments." *IAEA.*

Ito, M., and M. Suzuki. 2020. "Optimal maintenance scheduling in nuclear power plants via linear programming considering the relationship between the failure cause and maintenance type." *E-Journal of Advanced Mainte- nance* 11: 153 - 162.

Jiejuan, T., M. Dingyuan, and X. Dazhi. 2004. "A genetic algorithm solution for a nuclear power plant risk–cost maintenance model." *Nuclear Engineering and Design* 229: 81 - 89.

Lapa, C. M., C. M. Pereira, and A. C. d. A. Mol. 2000. "Maximization of a nuclear system availability through maintenance scheduling optimization using a genetic algorithm." *Nuclear engineering and design* 196(2): 219–231.

Martorell, S., S. Sanchez, S. Carlos, and V. Serradell. 2002. "Comparing effective- ness and efficiency in technical specifications and maintenance optimiza- tion." *Reliability Engineering & System Safety* 77: 281 - 289.

McCall, J. J. 1965. "Maintenance policies for stochastically failing equipment: a survey." *Management science* 11(5): 493–524.

Mollahassani-Pour, M., A. Abdollahi, and M. Rashidinejad. 2014. "Application of a novel cost reduction index to preventive maintenance scheduling." *International Journal of Electrical Power & Energy Systems* 56: 235 - 240.

Nilsson, J., A. Wojciechowski, A. Stromberg, M. Patriksson, and L. Bertling. 2009. "An opportunistic maintenance optimization model for shaft seals in feed-water pump systems in nuclear power plants." *2009 IEEE Bucharest PowerTech* 1 - 8.

Optimization, L. Gurobi. 2022. "Gurobi Optimizer Reference Manual."

Paz, N. M., and W. Leigh. 1994. "Maintenance scheduling: issues, results and research needs." *International Journal of Operations & Production Management.*

Peng, F., S. Kang, X. Li, Y. Ouyang, K. Somani, and D. Acharya. 2011. "A heuristic approach to the railroad track maintenance scheduling problem." *Computer-Aided Civil and Infrastructure Engineering* 26(2): 129-145.

Pereira, C. M., C. M. Lapa, A. C. d. A Mol, and A. F. da Luz. 2010. "A particle swarm optimization (pso) approach for non-periodic preventive maintenance scheduling programming." *Progress in Nuclear Energy* 52: 710 - 714.

Perez-Canto, S, and J. C. Rubio-Romero,. 2013. "A model for the preventive maintenance scheduling of power plants including wind farms." *Reliability Engineering & System Safety* (119): 67-75.

Samaranayake, P., and S. Kiridena. 2012. "Aircraft maintenance planning and scheduling: an integrated framework." *Journal of Quality in Maintenance Engineering.*

Sriram, C., and A. Haghani. 2003. "An optimization model for air-craft maintenance scheduling and re-assignment." *Transportation Research Part A: Policy and Practice* 37(1): 29-48.

Zhan, Z.-H., J. Li, J. Cao, J. Zhang, H. S.-H. Chung, and Y.-H. Shi. 2013. "Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems." *IEEE transactions on cybernetics* 43: 445 - 463.

Zhang, S., M. Du, J. Tong, and Y. F. Li. 2019. "Multi-objective optimization of maintenance program in multi-unit nuclear power plant sites." *Reliability Engineering & System Safety* 188: 532–548.

Zhong, S., A. A. Pantelous, M. Beer, and J. Zhou. 2018. "Constrained non-linear multi-objective optimisation of preventive maintenance scheduling for offshore wind farms." *Mechanical Systems and Signal Processing* 104: 347 – 369.

VITA

Timothy Virgil James Gallacher was born on May 11, 1987 to Claire and Kerry Gallacher. He attended Grace Brethren High School in Simi Valley, CA. Upon graduating High School, Tim enlisted in the U.S. Navy, where, after initial Nuclear Operator training, volunteered for the Submarine Service. Tim completed his qualifications as a Submariner in 2009, and immediately began working toward his undergraduate degree.

Tim graduated from Thomas Edison State University in 2012 with a Bachelors of Science in Nuclear Engineering Technology, and immediately enrolled in graduate school at the University of North Carolina at Charlotte. Tim graduated from UNCC in 2013 with a Masters of Science in Engineering Management, and one year later was honorably discharged from the US Navy, having served 8 years and 3 months.

Tim began employment with Exelon corporation upon separating from the Navy, in training to become a licensed Senior Reactor Operator. He successfully passed the SRO license exam with the NRC in 2017, and began overseeing operations at Limerick Generating Station.

In the Fall of 2019, Tim enrolled at the University of Tennessee, Knoxville, to pursue his PhD in Industrial Engineering under the guidance of James Ostrowski. He was supported in part by the University's partnership with NEUP at the Idaho National Lab. Tim completed his PhD in Industrial Engineering in December 2023.