3-22-2018

Scheduling Tool for the Nevada Test and Training Range

Miguel J. Macias

Scheduling Tool for the Nevada Test and Training Range

THESIS

Miguel J. Macias, 1st Lt, USAF

AFIT-ENS-MS-18-M-138

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-18-M-138

SCHEDULING TOOL FOR THE NEVADA TEST AND TRAINING RANGE

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Miguel J. Macias, B.S.

1st Lt, USAF

22 March 2018

AFIT-ENS-MS-18-M-138

SCHEDULING TOOL FOR THE NEVADA TEST AND TRAINING RANGE

THESIS

Miguel J. Macias, B.S.
1st Lt, USAF

Committee Membership:

Jeffery D. Weir, PhD
Chair

Lt Col Jeremy Jordan
Member

AFIT-ENS-MS-18-M-138

Abstract

Presently, the 57th Wing Scheduler at Nellis AFB schedules daily mission requests to the Nevada Test and Training Range (NTTR) airspace manually. The process is time consuming and may lead to suboptimal range resource allocations. The goal of this study is to provide the scheduler with an automated scheduling approach that will improve range scheduling efficiency. The tool developed uses range request data from units at Nellis AFB to produce daily mission schedules for a month long scheduling horizon with Microsoft VBA code and a commercial Integer Program (IP) solver. Under our current understanding of scheduler priorities, we formulate the problem with three different sets of objective function coefficients to maximize the utility of the schedules. We then analyze the differences in the schedules produced from a test data set comprised of requests from May 2017. Based on our analysis and results from testing, we recommend that the 57th Wing Scheduler employ one of our priority-based formulations to find fast and good feasible starting solutions to their monthly schedule build. This study demonstrates that using an IP approach to scheduling missions to the NTTR is feasible and has utility to the scheduling organization.

DEDICATION

To my Parents, who taught me the most important lessons in life and whose support sustains me.

To my Grandpa, whose example I always strive to follow.

Acknowledgements

I would like to express thanks to my family and friends for all their unconditional support in this endeavor. Additionally, I'd like to express my appreciation to my faculty research advisory, Dr. Weir, for all of his technical expertise and patience along the way. Finally, I want to thank Maj Brandon "Boar" Baughman and Maj Rolf "Nuke" Tellefsen of the 57 OSS at Nellis for assisting in this work and providing a realistic context for this study.

Miguel J. Macias

Contents

Page

List of Figures

List of Tables

SCHEDULING TOOL FOR THE NEVADA TEST AND TRAINING RANGE

I. Introduction

1.1 Background

A number of Department of Defense (DOD) test ranges are located on various military complexes throughout the United States. A wide variety of military systems are tested and evaluated within these specially designated tracts of land, water, and airspace. The largest and most sophisticated facility is the Nevada Test and Training Range (NTTR). A multitude of aircraft subsystems, such as electronic combat, navigation and guidance systems are tested and evaluated each day within this expansive test range complex. The range comprises 2.9 million acres of ground space and approximately 12,000 square nautical miles (NM) of airspace.

The organization at Nellis AFB responsible for managing range time allocation is 57th Wing Scheduling (57 OSS/OSOS). In performing this task, 57 OSS/OSOS manages the range, working in close collaboration with the entire spectrum of range users. A variety of aircraft and highly instrumented ground facilities must be coordinated to perform major tests and exercises on or above NTTR. One of 57 OSS/OSOSs primary functions, therefore, is to efficiently manage and schedule these assets to support a variety of requested missions. Each mission may require the use of several range areas and facilities, along with other resources such as instrumented aircraft, drones, high-powered radars, cameras, telemetry frequencies and airborne electronic countermeasure equipment.

1.2 Problem Statement

Presently, 57 OSS/OSOS produces daily flying schedules manually, using scheduler experience and knowledge to assign the missions to range block times. The mission schedules take fifteen hours to complete and must be finished six weeks in advance. The schedules are full of complexity and very difficult problems to solve, therefore suboptimal allocations are likely to occur. The goal of this thesis is to provide 57 OSS/OSOS with an automated scheduling approach that will improve range scheduling efficiency and flexibility. Past thesis work has been accomplished by Antes [1], McDaniel [2], and Liljenstolpe [3] on similar problems involving test mission scheduling at the Eglin AFB Test Range. Likewise, Hassel [4] and Foster [5] attempted to improve daily sortie scheduling for the USAF Test Pilot School (TPS). Through careful study of their work and lessons learned, we constructed a methodology that borrows and builds upon their efforts as we attempted to meet the needs of the Nellis Range community.

1.3 Approach

The objective of this thesis is to design and build an automated scheduling tool to aid the scheduler. We assign mission requests to range resources using three different approaches based on known scheduling priorities. The prescribed tool reads in requests using Microsoft VBA code, solves an integer program (IP) formulation using a mixed-integer program solver in MATLAB, then passes the solution back to VBA where a graphical solution gets generated. We will discuss all the assumptions, inputs, and the formulation to problem in depth in Chapter 3.

1.4 Summary

The paper is structured to review past work and concepts, explain our detailed methodology, and finally analyze the results of our work. Chapter 2 includes a discussion of past scheduling work done on DoD test ranges and the various optimization techniques that can be applied to solve the problem. From our review, we find that IP is the most appropriate technique given the problem size and has the potential for the best results. In Chapter 3, we outline the methodology used to define variables, parameters, and appropriate IP formulations. We discuss and analyze the solutions obtained from the IP solver in Chapter 4. Finally, Chapter 5 summarizes our findings, and makes recommendations for implementation and future work.

II. Literature Review

2.1 Scheduling

In general, a scheduling process involves the servicing of a fixed system of jobs (missions) by a set of resources (available range time, range areas, range assets, aircraft, etc.) over a given time period (one day), or as Pinedo states, "the allocation of resources to tasks over given time periods with the goal of optimizing one or more objectives" [6]. Within a scheduling process scoped for the needs of Nellis, the goal of a scheduling algorithm may be to maximize the number of missions scheduled on any given day. Likewise, the goal may be to minimize the maximum tardiness of a mission that fulfills a test or syllabus requirement. In this thesis, we maximize the number of missions scheduled daily for a month long scheduling horizon.

In general, the fulfillment of the desired objective using an automated scheduling approach would improve range scheduling efficiency, the primary goal of this thesis. At this time, no practical algorithms or procedures are known to have been developed to accommodate range scheduling specifically for the NTTR. However, it appears that efforts have been made by Antes [1], McDaniel [2], and Liljenstolpe [3] to improve scheduling at other test ranges within the DOD, particularly the Eglin Test Range. A common theme found among their papers is their identification of the problem as one of a special class of scheduling problems, Resource Constrained Scheduling (RCS). Applying an analogous problem solving technique to the Nellis problem has the potential to yield the most insightful results.

Resource Constrained Scheduling (RCS) Problem

Resource Constrained Scheduling (RCS) refers to scheduling problems that deal with scheduling activities under limited time and resource constraints. Walker and

Chaudhuri define RCS problems using the following criteria:

> **Given:**
>
> A set O of operations; a set K of functional unit types; a type function: $O \rightarrow K$; resource constraints m_k, $1 \leq k \leq K$ for each functional unit type; and a partial order on O determined by the precedence constraints.
>
> **Find:**
>
> A feasible (or optimal) schedule for O that obeys the precedence constraints and that meets the resource constraints for each functional unit [7].

These types of problems are surrounded by enormous computational complexities which has earned them the classification of NP-complete. Common to such problems are activities of known duration that need to be scheduled within a certain time frame, along with predetermined levels of resources that are limited in quantity [2]. The daily scheduling of test and training missions and their requested resources at Nellis fall under this category of scheduling. In the following sections we will explore solution techniques used in the past to solve these types of problems.

2.2 Heuristics

Heuristics have become an important area of research and application within optimization. Their appeal stems from their ability to quickly produce high quality solutions to difficult optimization problems. As opposed to the advanced mathematical proofs that are required to develop theoretical results in optimization, the development of heuristics is chiefly an art and a creative problem solving endeavor [8]. A plethora of conceptual approaches to these algorithms exist within the field. In 1989, Zanakis, Evans, and Vazacopulos categorized the algorithms from 442 pub-

lished articles into 12 separate classes. Of the 442 papers cited, they designate 112 as directly applicable to solving scheduling problems. Among the heuristics discussed in the paper is Tabu Search, a relaxation algorithm created by Fred Glover that is one of the most cited algorithms in the field due to its exceptional quality [9].

Tabu Search

Tabu Search is a strategy for solving combinatorial optimization problems whose applications range from scheduling and computer channel balancing to cluster analysis and space planning. Research and computational comparisons have disclosed the ability of Tabu Search to obtain high quality solutions with modest computational effort [9]. Sarkheyli, Bagheri, Ghorbani-Vaghei, and Askari-Moghadam constructed a variation of Tabu Search to schedule Low Earth Orbit (LEO) satellite missions with the objective of maximizing the number of scheduled tasks [10]. The Tabu Search algorithm with a modified move operation produced near-optimal solutions with less computational effort than a Genetic Algorithm and a Hill Climbing Algorithm. Given Tabu Search's reputation and the promise it shows as a high quality heuristic search algorithm, we considered it to be a strong candidate tool that could be applied to the Nellis problem.

Greedy Randomized Adaptive Search Procedure (GRASP)

Another promising heuristic search technique to solve difficult combinatorial optimization problems is a Greedy Randomized Adaptive Search Procedure (GRASP). The algorithm uses a combination of random and greedy elements to construct and improve upon solutions in the search space. The algorithm maintains a restricted candidate list (RCL), in which it stores high quality solutions, to guide the search into more promising neighborhoods where the optimal solution may be found [11].

Erdemir completed past scheduling thesis work using a GRASP algorithm within his methodology [12]. He found that the GRASP algorithm produced quick and efficient solutions to his fighter squadron scheduling problem. Though GRASP is generally not as robust as other heuristics such as Tabu Search, it could prove to be a quick, easy, and highly implementable way to solve the Nellis problem should programming limitations arise.

Unique Heuristic Approaches

Though not easily categorized within the field of heuristics, past work on range scheduling has produced unique heuristic solution methods. Antes' follow-on work of Hallis' paper used constructive heuristics to develop an interactive scheduling algorithm for the Eglin Test Range scheduling problem [1]. He designed the new algorithm for integration into a preexisting scheduling management system, RESOMS. The algorithm sought to schedule missions according to critical resource groups while ensuring mission priority was not violated. The algorithm saw some improvement over previous methods used for scheduling at Eglin but required man-in-the-loop intervention to guide the heuristic search.

Liljenstolpe's work on the same problem 17 years later produced a single pass scheduling algorithm that employed greedy methods to schedule missions based on priority. The algorithm produced feasible solutions but failed to utilize all available range time and manpower. As a result, Lilijenstolpe encouraged the implementation of an improvement metaheuristic for follow-on work [3]. While heuristics offer attractive good solutions in a reasonable amount of time, they do not guarantee optimal solutions, therefore they should only be used when optimal solution techniques fail. Since no work currently exists on the Nellis problem, we had to consider the employment of an integer program technique to solve the problem before trying a

heuristic.

2.3 Integer Programming

An integer program (IP) is a form of a linear program (LP) in which some or all of the decision variables are required to be non-negative integer values [13]. Many real-life problems, such as the Nellis problem, require the decision variables to take on integer values. Due to the additional integer constraint found in an IP, it is usually much harder to solve than an LP. Depending on the problem, it maybe permissible to solve the problem using an LP and then round the answer. In other cases, this practice is not acceptable, therefore a specific IP solution technique is required. Extensive work has been done in this field of study that encompasses three subfields: pure integer programming [14], binary integer programming [15], and mixed integer programming [16]. We determined that the structure of the Nellis problem lends itself to a binary integer program so we focused our effort on investigating solutions methods within this realm.

Binary Integer Programming

A plethora of integer programming methods to solve Generalized Assignment Problems (GAP) exist. At the introductory level, Bazaraa presents (Kuhns) Hungarian Algorithm as a method to solve a special version of a GAP, the transportation problem, when the number of resources are equal to the number of tasks. Within the Hungarian Algorithm, we wish to find the minimal cost assignment or a one-to-one matching of individuals to jobs [17]. Though the Hungarian Algorithm is a proven and reliable algorithm for solving an assignment problem, it is not appropriate for the structure of the Nellis problem.

The Lagrangian relaxation method is another approach to solving an assignment

problem via integer programming. Fisher writes that there exist two natural Lagrangian relaxations for the assignment problem. The first is obtained by dualizing the assignment constraint, while the second involves dualizing the resource and binary decision variable constraints. Fisher's methods attractively offer solutions obtainable in O(mn) polynomial time [18]. Should computational time become a serious problem when solving the Nellis problem, Lagrangian relaxation could be applied.

A practical application of an integer program used to solve an assignment problem can be found in Hassel's work on TPS Scheduling [4]. After employing decision variable reduction rules, she used a branch-and-bound algorithm to solve the problem. She found that her method could produce a feasible schedule for small problems that were representative of a portion on the entire TPS scheduling, but could not provide solutions to the full TPS problem.

Though Hassel was unsuccessful in her efforts to solve the entire large scale problem, the proven ability of branch-and-bound appeared to be a promising solution technique that could be applied to the Nellis problem. Additionally, the technique is found in many commercially available solvers, which provided for ease of implementation. We later found that branch-and-bound alone was not sufficient due to the demand for range resources being far greater than the supply of available resources. Due to this shortage, a priority based approach to the order in which missions were scheduled had to be employed, therefore we examined preemptive goal programming and how its concepts could be applied to the Nellis problem.

2.4 Preemptive Goal Programming

Preemptive goal programming can be applied to an optimization problem when a decision maker is not able to precisely determine the relative importance of the goals [13]. To apply preemptive goal programming, the decision maker must simply

rank his or her goals from most important to least important. The program will then try to satisfy the first goal before it moves onto lower priority goals. For the Nellis problem, resource limitations prevent us from scheduling all missions, therefore we must prioritize the order in which particular types of missions get scheduled. Scheduling missions with high priority Test Priority List (TPL) values are of the utmost importance. Additionally, Fighter Weapons School (WPS) are high priority missions since they must adhere to syllabus requirements. Given the distinction of varying priorities for different mission types, implementing a goal program like approach to schedule missions based on their priority group was appropriate for the Nellis problem.

III. Methodology

3.1 Overview

The NTTR Scheduling Tool uses Nellis range request data to produce daily mission schedules for a month long scheduling horizon. The tool, programmed using Microsoft Visual Basic Application (VBA), reads, cleans, and sorts the data to build an IP formulation out of the daily mission requests. The formulation gets passed to a commercial solver, intlinprog in MATLAB, which searches for feasible solutions as missions get added to a daily schedule. When the solver finds an optimal solution, maximizing the objective function, or a feasible solution obtained within a time limit, the solution gets passed back to the tool, where a graphical representation of the daily schedule gets produced. Figure 1 displays the process the tool employs in a flow chart. We will discuss the data source, the assumptions applied to the problem, and the formulation for the problem in the proceeding sections.

3.2 Data Source

The data source used to schedule missions to the NTTR is comprised of range requests from the units at Nellis who use the range. Units submit requests in the form of Microsoft Excel workbooks. Figure 2 displays an example of a range request worksheet. The range requests include data elements such as date desired, time of day desired, duration, type of mission, the missions' TPL designation, ordnance to be used, the number of blue air participants, the number of red (adversary) air participants, and the subranges required to carry out the mission. The requesting units can also include remarks for special requests outside of the standard request protocol.

For the sake of this scheduling endeavor, we identified date, time of day, duration,

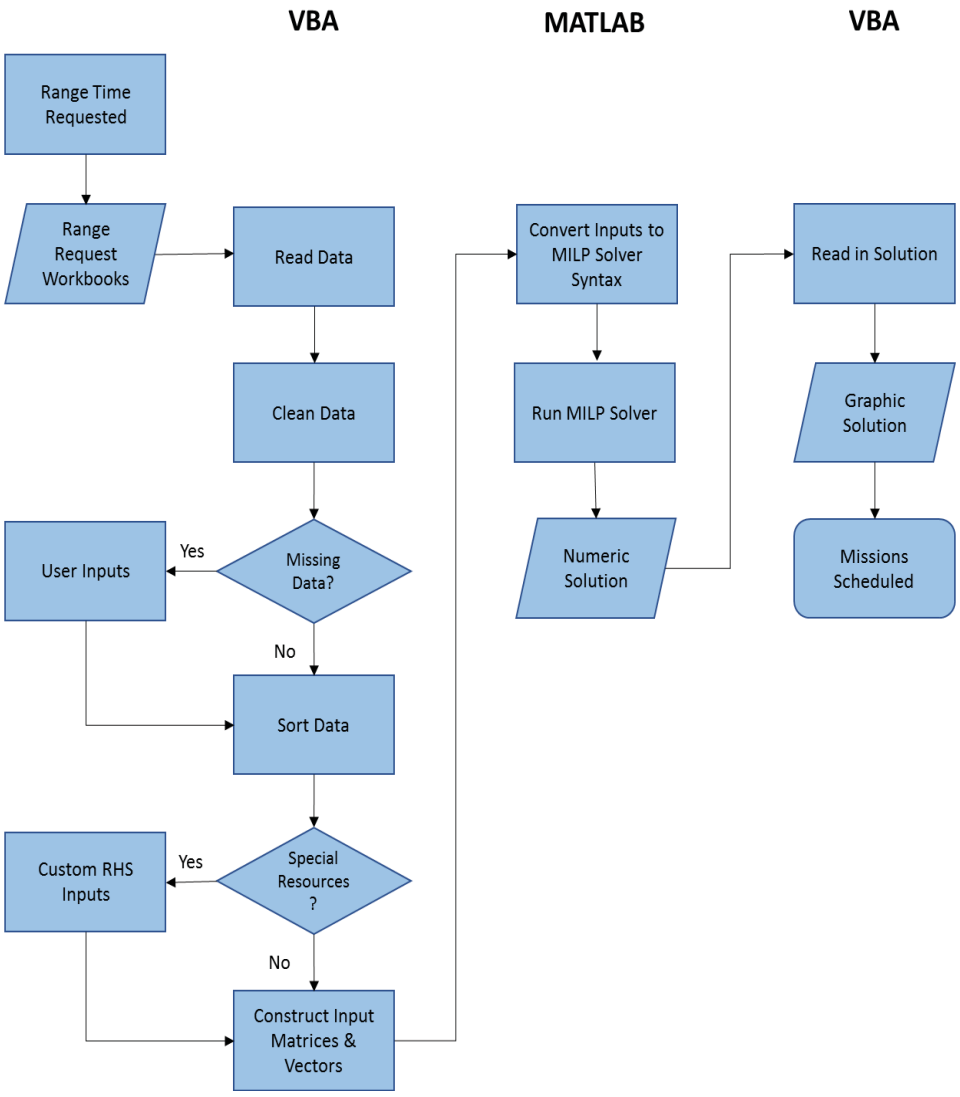

Figure 1. *Scheduling Tool Process*

and Test Priority List (TPL) number as key data elements to schedule requested missions to subranges within a discrete set of time blocks. We read data from each units range request using VBA to extract the previously mentioned data elements for all missions.

NELLIS AFB RANGE REQUEST FOR							May-17							422 F-15E
DATE OF REQUEST:				[redacted]	POC:		[redacted] POC Voice: [redacted] POC EMAIL: [redacted]							
DATE DUE:				[redacted]										
MISSION SPECIFICS							# AC	ADV	TUR	71N 71S 76 ECS PMB PMA O8B O8C O8D O8E ECW ECE O9A 75W 75E 74A 74B 74C CYA CYB CYC CYD RMS RMN	CAA CAB CAC ELG	SAL	ALA ALB ALC 61A 61B 62A 62B 65A 65B 65C 65D 64A 64B 64C 64D 64E 64F 63B 63H 63C	REMARKS
FROM	TO	TIME	DUR	TYPE OF MISSION	TPL #	ORD								
5/1-Mon	5/4-Thu	DAY	1.5	[redacted]	58	C/F	2	N	3.0	X	X			
5/5-Fri	5/5-Fri	DAY	1.5	[redacted]	58	C/F	2	N	3.0	X	X			
5/8-Mon	5/11-Thu	DAY	1.5	[redacted]	58	C/F	2	N	3.0	X	X			
5/12-Fri	5/12-Fri	DAY	1.5	[redacted]	58	C/F	2	N	3.0	X	X			
5/15-Mon	5/18-Thu	DAY	1.5	[redacted]	58	C/F	2	N	3.0	X	X			
5/19-Fri	5/19-Fri	DAY	1.5	[redacted]	58	C/F	2	N	3.0	X	X			
5/22-Mon	5/25-Thu	DAY	1.5	[redacted]	58	C/F	2	N	3.0	X	X			
5/26-Fri	5/26-Fri	DAY	1.5	[redacted]	58	C/F	2	N	3.0	X	X			
5/29-Mon	5/31-Wed	DAY	1.5	[redacted]	58	C/F	2	N	3.0	X	X			

Figure 2. *Sample Range Request Inputs*

3.3 Assumptions

The assumptions made in the problem can be traced back to the nature of the inputs and the chosen solution method. To correct for shortcomings in the input data, we made assumptions about the nature of non-standard input formats in the date, time of day, duration, and subrange requested data. The chosen solution technique, integer programming, forced us to discretize the time element of the problem into 96 defined 15-minute time blocks distributed over the course of a 24 hour period. These assumptions, which we will discuss in detail below, shaped our prescribed formulation for the problem.

Typically, a requesting unit designates a range of dates for which a mission will be flown. We interpreted this range as a set of days in which one mission will be flown on each day included in the set. In some cases, the raw data did not include a continuous range of days. A user input is required to clarify that the range of days was not intended to be continuous and break out the requests into multiple single day or a set of multi-day continuous requests. We assume that the intended user can decipher the true meaning of and correct the non-standard date inputs that cannot be directly read by the existing code.

The time of day data in each request takes on two different types, numeric and string. Some units prefer to designate Day or Night, AM or PM as sufficient requirements for the time of day in which their mission must be flown, while others prefer to use more specific numeric times, such as 1200-1500 to define the time window in which they must complete their mission. We developed logic within the code that translates all mission times of day into numeric time windows in which each mission must be flown. For example, a mission designated as a Day mission must start no earlier than sunrise and end no later than sunset, while a Night mission must start and end before sunrise, or start and end after sunset for the designated day. For ease of coding, sunrise is defined as 0600 and sunset is defined as 1830 in this thesis. Similarly, a mission designated as an AM mission must start and end between 0000 and 1200, while a mission designated as a PM mission must start and end between 1200 and 2400.

The duration data in each request specifies the amount of time the mission plans to last. Some duration requests specify duration time in hours, using decimal values to indicate fractional hours, while others specify the time in hours and minutes. If the amount of time requested is not divisible into an integer number of 15-minute blocks then the nearest number of 15-minute blocks required to fulfill their mission

requirements is rounded up. Additionally, some durations times are missing in the requests. In this case, we assume the missions last exactly one hour.

Finally, we assume that a mission gets assigned the subranges requested at all possible flight levels throughout the entire duration of the mission. In reality, there may be cases in which more than one user can use range space at the same time and the requesting units would deconflict based on their flight levels. We programmed a user interface that will allow the user to adjust the number of missions present in a subrange at a specified time. The interface adjusts the RHS in the formulation, adding resources to allow for a looser constraint.

3.4 Formulation

We formulated the problem as a maximization problem in which we attempt to schedule as many missions as possible to a daily schedule subject to resource and time constraints. We prescribe two different formulations to the problem, one that lends itself to a general form of the problem, the initial formulation, and another that lends itself to the problem being solved in MATLAB, since MATLAB solvers only accommodate formulations in which the decision variables and right hand side (RHS) of the constraints are stored in vectors. We modified the MATLAB formulation to explore scheduling missions using three different objective function coefficient sets to accommodate known scheduler priorities.

Matrix Indices

The initial formulation involved three sets of indices: mission active time (t), subrange (j), and mission number (k). The MATLAB formulation involves four sets of indices: mission active time (t), mission start time (u), subrange (j), and mission number (k).

Decision Variables

The initial formulation of the problem involved the construction of a matrix in which we could identify which mission was active during each discrete time block. We defined the decision variable matrix, X, as a k by t matrix, where k is the number of missions to be scheduled each day and t is the number of discrete time blocks in the problem. A one in the matrix indicates that mission k will be active at time t, while a zero indicates that the mission will not be active. Additionally, we use a binary indicator decision variable, y_k, that tells us if mission k gets scheduled or not. We used the variable in constraint set three to model a logical situation.

Due to the input argument format required by intlinprog in MATLAB, we converted the decision variable matrix into a single vector, x, and redefined the indices as k, u where k is the mission number and u is the starting time of mission k. A one in index k, u indicates that mission k begins at time u, while a zero indicates the mission does not start at that time.

Objective Function

For the initial formulation, the objective function consists of the sum of all the elements in the decision variable, while the MATLAB formulation uses the sum of the product of the decision variable vector components and their associated coefficients.

When solving for the maximum number of missions in MATLAB, we assume all missions contribute equal value to the objective function, so all coefficients are equal to one in this case. When solving for a priority based solution in MATLAB, the coefficients associated with each mission are equivalent to 2^{n-k} where n is the total number of missions requested each day and k is the priority index of the associated mission. As k increases, the relative priority of the associated mission decreases.

Constraint Set 1

The first set of constraints in the formulation ensure no more range resources are consumed than are made available. In the problem, the resources are the combination of a subrange and a 15-minute time block. In the initial formulation, we identified that resource consumption had to be less than or equal to the resources available. In this case, resource consumption is the product of the A matrix, which consists of the subranges requested given a mission, and the decision variable matrix, X. We stored the resources available in the B matrix. Under our assumption that only one mission can be active in a subrange at each discrete point in time, all elements in the B matrix are equal to one. Since the constraint set in the initial formulation does not lend itself to a form that can be read by intlinprog, we modified the formulation.

The A matrix in the MATLAB formulation consists of generated binary patterns based on times the mission will be active in the subrange given a starting time. The column indices of A, u, k represent the starting time and mission number respectively, while the row indices j, u represent the subrange and time the mission is active in the subrange based on the starting time, respectively. Given a column index k, u, row indices j, u, $j, u+1$,...$j, u+d_k-1$, where d_k is the duration of mission k in number of 15 minute intervals, get populated with the corresponding range request pattern for each mission. The pattern of ones and zeros for a block j, u for $j = 1, 2, ...52$ is based on the subranges requested for each mission in the range request workbooks. If a subrange is requested, the corresponding j index gets a one, while if it is not requested the index gets a zero. The RHS of the constraint, $b_{j,t}$, represents the number of resources available in each subrange at each 15-minute time block.

Constraint Set 2

The second set of constraints ensure the missions begin within the constrained time frame. The original formulation involved a comparison of the elements in the decision variable matrix, X, and the T matrix, which represents the limitation on the time of day in which a mission gets completed. We derived the T matrix from the time of day data to determine all the 15-minute time blocks in which each mission could be active. The rows of T, k, represent the missions requested for the day while the columns, t represent a discrete set of times ranging from 0000-2400, separated by 15 minutes. We represent feasible active mission times with ones in the matrix and infeasible times with zeros. To satisfy this constraint set, an element in the X matrix must be less than the corresponding element in the T matrix.

In the MATLAB formulation, we defined a new matrix, J, and derived the information it contains from the T matrix. Using the T matrix, we defined the earliest possible mission start time and the latest possible mission start time. The earliest possible mission start is the row index in which the first one appeared in each row (mission). We found the latest possible start time by identifying the greatest column index in which a one appeared and subtracted the duration of the mission from that index. We transcribed the earliest start and latest start time for each mission into the J matrix. The row indices, k, represent the mission number, while the column indices k, u , represent mission number and mission starting time. We placed a one in all k, u indices in which mission k can begin. For example if the intersection of row index 1 and column index 1, 33 contains a one, then mission 1 is allowed to begin at time 0800. The RHS of the inequality constraint set, e, contains a column vector of ones of size k, which ensures no more than one starting time is chosen for each mission.

Constraint Set 3

The third set of constraints in the initial formulation ensures that each scheduled mission is active for the duration requested. We stored all the requested mission durations in a vector, d_k, where each duration is expressed in an integer number of desired 15-minute blocks. The left hand side assesses the sum of the number of 15-minute blocks each mission occupies, while the right hand side specifies a duration required. We model a logical situation in which we determine if mission k gets scheduled or not using an indicator decision variable, y_k. When mission k is not scheduled, y_k is equal to zero and it forces the right hand side to equal zero, thereby forcing column k of the decision variable matrix to be populated with zeros.

In the MATLAB formulation, the H matrix ensures that if a mission gets scheduled, it occurs at an allowable start time. We structured the H matrix in a similar manner to the J matrix, however wherever a one appeared in the J matrix, a zero appears in the H matrix and wherever a zero appeared in the J matrix, a one appears in the H matrix. The ones appear in indices that represent disallowed mission start times, while the zeros appear in indices that represent allowable mission start time. The right hand side of the equality constraints contain a vector of zeros such that if the IP chose a disallowed start time the constraint would be violated.

Initial Formulation

$$X_{k,t} = \begin{cases} 1 \text{ if mission } k \text{ is active at time } t \\ 0 \text{ otherwise} \end{cases}$$

$$T_{k,t} = \begin{cases} 1 \text{ if mission } k \text{ can be active at time } t \\ 0 \text{ otherwise} \end{cases}$$

$$A_{j,k} = \begin{cases} 1 \text{ if mission } k \text{ requires subrange } j \\ 0 \text{ otherwise} \end{cases}$$

$$B_{j,t} = \begin{cases} 1 \text{ if subrange } j \text{ is available at time } t \\ 0 \text{ otherwise} \end{cases}$$

$$y_k = \begin{cases} 1 \text{ if mission } k \text{ is scheduled} \\ 0 \text{ otherwise} \end{cases}$$

$d_k =$ duration of mission k in number of 15 minute intervals

$t = 1, 2...96$

$k = 1, 2...$number of missions

$j = 1, 2...52$

$$max \sum_k \sum_t X_{k,t} \tag{1}$$

$$A_{j,k} X_{k,t} \leq B_{j,t} \qquad \forall\ j, t \tag{2}$$

$$X_{k,t} \leq T_{k,t} \qquad \forall\ k, t \tag{3}$$

$$\sum_t X_{k,t} = d_k y_k \qquad \forall\ k \tag{4}$$

$$X_{k,t} \in \{0, 1\} \qquad \forall\ k, t \tag{5}$$

$$y_k \in \{0, 1\} \qquad \forall\ k, \tag{6}$$

MATLAB Formulation

$$x_{ku} = \begin{cases} \\ \text{1 if mission } k \text{ starts at time } u \\ \\ \text{0 otherwise} \end{cases}$$

$$A_{jt,ku} = \begin{cases} \\ \text{1 if mission } k \text{ will be active in subrange } j \text{ at time } t \text{ if the mission begins at time } u \\ \\ \text{0 otherwise} \end{cases}$$

$$J_{k,ku} = \begin{cases} \\ \text{1 if mission } k \text{ is allowed to start at time } u \\ \\ \text{0 otherwise} \end{cases}$$

$$H_{k,ku} = \begin{cases} \\ \text{0 if mission } k \text{ is allowed to start at time } u \\ \\ \text{1 otherwise} \end{cases}$$

c_{ku} = objective function value contribution if mission k starts at time u

b_{jt} = number of resources available in subrange j at time t

e_k = vector of ones of size k

f_k = vector of zeros of size k

d_k = duration of mission k in number of 15 minute intervals

$t = 1, 2...96$

$u = 1, 2...96$

$k = 1, 2...$number of missions

$j = 1, 2...52$

$$max \sum_{ku} c_{ku} x_{ku} \tag{7}$$

$$A_{jt,ku}\ x_{ku} \leq b_{jt} \qquad \forall\ jt \tag{8}$$

$$J_{k,ku}\ x_{ku} \leq e_k \qquad \forall\ k \tag{9}$$

$$H_{k,ku}\ x_{ku} = f_k \qquad \forall\ k \tag{10}$$

$$x_{ku} \in \{0, 1\} \qquad \forall\ ku \tag{11}$$

3.5 Scheduling Methods

We formulated the problem using three different sets of objective function coefficients and then identified which set yielded the best schedules based on performance measures that we will discuss in detail in Chapter 4. The first set attempts to schedules as many missions as possible, regardless of mission affiliation. In this formulation, all objective function coefficient values are equivalent, which allows us to find the maximum number of scheduled missions contained in a feasible solution for each day. We used these solutions as baselines for our two priority based formulations.

Resource limitations prevented us from scheduling all missions at once, therefore the second and third methods use priority driven objective function structures to incentivize the scheduling of higher priority missions over lower priority missions. Since test programs have strict test completion deadlines they must adhere to, scheduling Test Priority List (TPL) missions is of the utmost importance. Additionally, the USAF Fighter Weapons School (WPS) must adhere to syllabus time lines, therefore scheduling WPS missions is also of high priority. Our priority based IP formulations assigned the greatest objective function coefficient values to TPL missions. We assigned all WPS missions objective coefficient values that were less than all TPL mission coefficients but greater than any non-affiliated missions. Finally, all remaining missions received objective function coefficient values based on their duration.

3.6 Programming Considerations

Rationale for Using VBA and MATLAB

We used VBA and MATLAB as the two programming languages to build the scheduling tool. The use of VBA was necessary since the existing operating procedure of requesting range time involves the submission of Microsoft Excel workbooks that contain each units mission plan for a month to the Wing Scheduler at Nellis. These workbooks contain all the data necessary to automate the scheduling process in an existing loosely standardized format. Additionally, scheduling is not the Wing Schedulers primary duty, therefore using a program with which the scheduler is familiar facilitates the process. Taking this into consideration, we favored Microsoft Office programs over other programming languages. Figure 3 displays Microsoft Excel output from the tool that the scheduler can review before finalizing schedules.

Where VBA failed to meet the requirements of the project is in its ability to handle large scale optimization problems. The optimization tool pack in the standard version of Excel can only handle problems with up to 200 decision variables and up to 100 constraints. Due to this limitation, it was necessary to find a commercially available solver that could handle a larger scale formulation. MATLABs built-in Integer Program solver (intlinprog) meets the required needs. Additionally, VBA offers a project library add-in that can execute commands in and pass variables to MATLAB. The ease of interoperability between the two programs and meeting of project requirements justified the use of the two programs.

Algorithm

MATLAB's intlinprog uses a strategy with multiple IP solution techniques to solve the problem citeMATLABintlinprog. The solver can solve the problem in any of the stages. If it solves the problem in a stage, intlinprog does not execute the later stages.

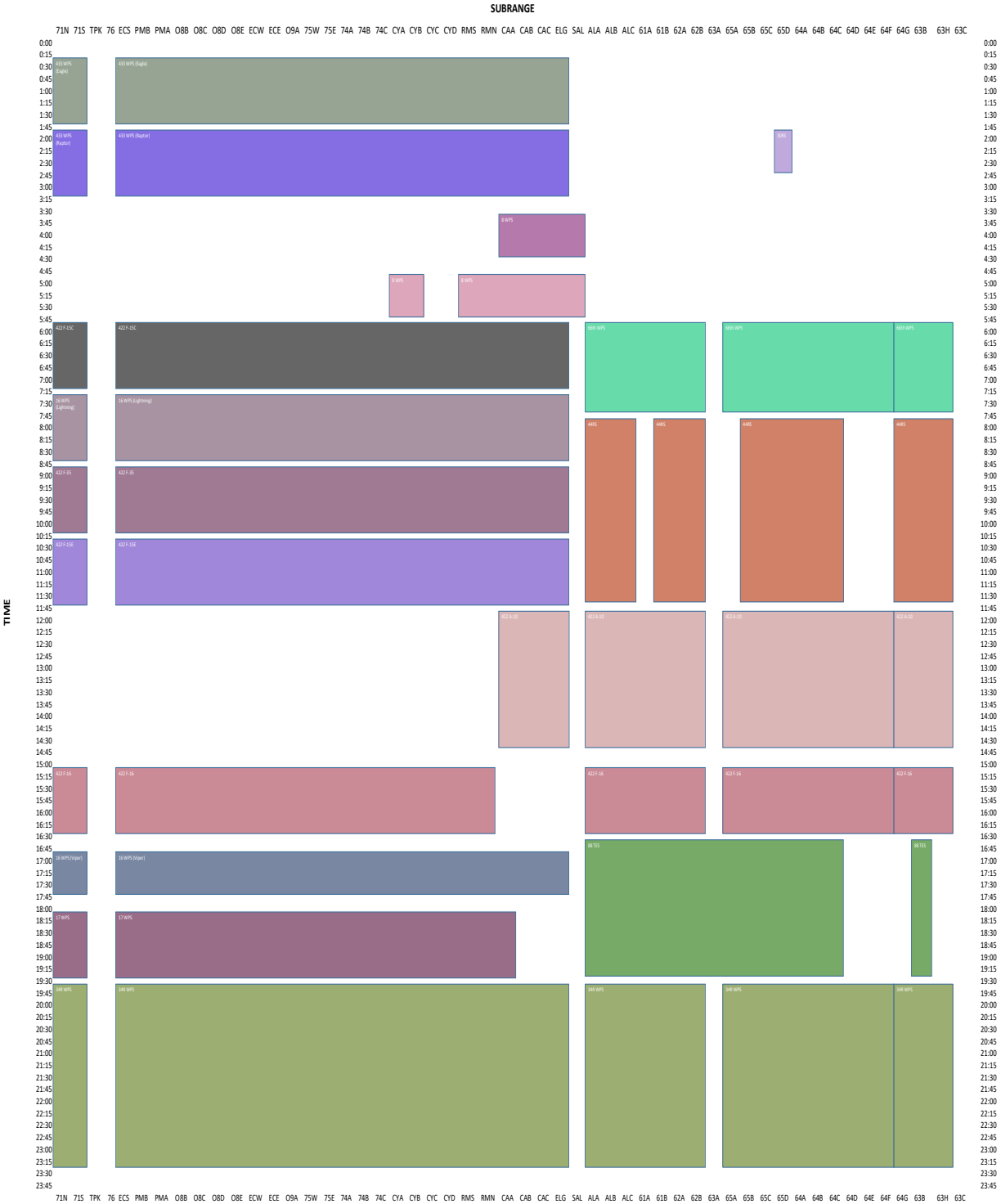

Figure 3. *Example Output*

The following list outlines the strategy:

1. Reduce the problem size using Linear Program Preprocessing.
2. Solve an initial relaxed (noninteger) problem using Linear Programming.
3. Perform Mixed-Integer Program Preprocessing to tighten the LP relaxation of the mixed-integer problem.
4. Try Cut Generation to further tighten the LP relaxation of the mixed-integer problem.
5. Try to find integer-feasible solutions using heuristics.
6. Use a Branch and Bound algorithm to search systematically for the optimal solution. This algorithm solves LP relaxations with restricted ranges of possible values of the integer variables. It attempts to generate a sequence of updated bounds on the optimal objective function value.

3.7 Conclusion

This chapter has discussed the methodology built to schedule requested missions to range resources. We apply four assumptions to the problem to correct for shortcomings in the input data. After applying our assumptions, we construct a general and MATLAB formulation for use in an mixed integer linear program solver. We prescribe three different sets of objective function coefficients to account for known scheduling priorities. We will discuss these objective function coefficients and the scheduling results in more detail in Chapter 4.

IV. Analysis & Results

4.1 Overview

This chapter outlines the results of the study. We discuss the use of data set to test the methodology, the three different sets of objective function coefficients used to formulate the problem, and the performance of schedules built under three different scenarios.

4.2 Range Requests

The data used to test the formulation of the problem consists of a month's worth of daily range requests for May 2017 from ten units. We identified the number of missions scheduled, number of TPL missions scheduled, number of WPS missions scheduled, and the number of resources scheduled as key performance measures. We summarized the daily missions requests in Table 1.

4.3 Method Comparison

We formulated the problem using three different sets of objective function coefficients and then analyzed the difference in the schedules produced by each formulation. For the sake of being able to easily reference the formulations, we assigned each objective coefficient formulation a name: Max Mission, Priority, and Priority Relaxation. Table 2 identifies the coefficients used for each formulation type. The Max Mission formulation schedules the maximum number of missions to a daily schedule regardless of mission affiliation. We assigned each mission starting time an objective coefficient value equal to one for this formulation.

The Priority formulation weights the missions based on their relative priority with the highest priority missions being given more weight. We used a 2^{n-k} objective

Table 1. Daily Requests

Date	Missions Requested	Resources Requested	TPL Requested	WPS Requested
5/1/2017	23	5627	4	12
5/2/2017	26	5539	4	11
5/3/2017	25	6043	4	12
5/4/2017	23	5707	4	12
5/5/2017	18	4569	3	7
5/8/2017	17	4107	4	6
5/9/2017	21	4799	4	6
5/10/2017	18	4527	4	6
5/11/2017	18	4527	4	6
5/12/2017	16	3865	3	5
5/15/2017	16	2910	4	6
5/16/2017	21	3794	4	6
5/17/2017	17	3330	4	6
5/18/2017	17	3418	4	6
5/19/2017	15	2964	3	5
5/22/2017	23	3846	4	7
5/23/2017	27	5023	4	7
5/24/2017	25	4567	4	7
5/25/2017	23	3747	4	7
5/26/2017	11	2164	3	5
5/27/2017	3	436	0	3
5/28/2017	7	716	0	3
5/29/2017	12	2050	4	4
5/30/2017	26	4101	4	7
5/31/2017	27	5802	4	7
6/1/2017	1	162	0	0
6/2/2017	1	162	0	0
Total	477	98496	88	169
Average	17.67	3648	3.26	6.26

coefficient set assigned to the decision variables corresponding to the possible mission start times for mission k. We sorted missions $k = 1...n$ in ascending order of their TPL number, their WPS affiliation, and finally their ascending duration. The Priority objective coefficient set works in a way such that the value of scheduling mission k is greater than the sum value of scheduling missions $k+1...n$. Under the realization that this value set could lead to schedules dominated by a single mission, we developed an alternative value approach to prevent this.

The Priority Relaxation formulation employs a $max(2^{n-k-2}, 1)$ objective function coefficient set instead. This method works in such a way that the value of scheduling mission k is one less than the sum value of scheduling missions $k+1...n$. This objective function formulation can be generalized to $max(2^{n-k-p}, 1)$, where as p increases, the relative priority ranking difference of each job decreases.

To analyze the differences between the three formulations, we compared their scheduling performance under the assumption that only one mission can be assigned to a subrange at any given time. We compared the number of missions scheduled, resources scheduled, Branch-and-Bound nodes explored, and the amount of time needed to find solutions. To confirm any differences between the scheduling formulations, we used a paired t-Test with $\alpha = 0.05$ to compare the number of missions scheduled for each day. Table 3 summarizes the paired t-test p-values that we will build upon in this section.

Table 2. Objective Function Coefficients

	Max Mission	**Priority**	**Priority Relaxation**
Coefficient	$1 \quad \forall k$	$2^{n-k} \quad \forall k$	$max(1, 2^{n-k-2}) \quad \forall k$

Table 3. Paired t-Test p Values

	Max Mission v. Priority	Max Mission v. Priority Relax.	Priority v. Priority Relax.
Missions Scheduled	0.162	0.162	0.000
TPL Scheduled	0.031	0.031	0.000
WPS Scheduled	0.327	0.327	0.000
Resources Scheduled	0.013	0.011	0.935
Run Time	0.006	0.157	0.027
B & B Nodes	0.013	0.139	0.039

Missions Scheduled

When assessing the number of missions scheduled, we found that the Max Mission formulation scheduled the most missions, but failed to schedule critical missions that the Priority and Priority Relaxation formulations scheduled. Table 4 summarizes the daily build by formulation type. Both priority based formulations produced two fewer total scheduled missions than the Max Mission formulation, however both priority based formulations produced statistically significant more scheduled TPL missions. The Priority formulation produced the best results with respect to the highest priority missions, scheduling all TPL missions, and 160 of 169 WPS missions. Table 5 summarizes these results. The underlying trade off with respect to missions scheduled is dependent on the value a decision maker places on a TPL mission, a WPS mission, and lower priority missions. A decision maker can have two more total missions at the cost of six fewer TPL missions and one fewer WPS mission scheduled.

We also analyzed the performance of the three formulations under the conditions that the resources requested for the day were less than or equal the total resources available. We define a resource as the combination of a 15-minute time block and a subrange. Since there are 96 time blocks and 52 subranges that comprise the schedule space, the total number of resources available when we maintain the assumption that only one mission can be scheduled to one range resource is 4,992. Six days of requests

Table 4. Missions Scheduled

Date	Max Mission	Priority	Priority Relaxation
5/1/2017	17	17	17
5/2/2017	19	19	19
5/3/2017	18	17	17
5/4/2017	17	17	17
5/5/2017	15	15	15
5/8/2017	14	14	14
5/9/2017	17	17	17
5/10/2017	15	15	15
5/11/2017	15	15	15
5/12/2017	13	13	13
5/15/2017	14	14	14
5/16/2017	17	17	17
5/17/2017	14	14	14
5/18/2017	14	14	14
5/19/2017	13	13	13
5/22/2017	19	19	19
5/23/2017	22	22	22
5/24/2017	21	21	21
5/25/2017	19	19	19
5/26/2017	10	10	10
5/27/2017	3	3	3
5/28/2017	7	7	7
5/29/2017	11	10	10
5/30/2017	21	21	21
5/31/2017	21	21	21
6/1/2017	1	1	1
6/2/2017	1	1	1
Total	388	386	386
Average	14.37	14.30	14.30
Total | Resources Requested $\leq$ 4992	274	273	273
Average | Resources Requested $\leq$ 4992	13.05	13.00	13.00

Table 5. Percent of Requested Missions Scheduled

	Max Mission	Priority	Priority Relaxation
All Missions	81.3%	80.9%	80.9%
TPL	93.2%	100.0%	100.0%
WPS	94.1%	94.7%	94.7%

exceed the 4,992 resource threshold, therefore the results for those days are excluded from the results reported in Table 6. The total percentage of missions scheduled increases for all three formulations. We obtained identical results for both priority formulations. Neither of the three formulations produced statistically significant different results under this condition. A single trade off in which a decision maker can have one more mission scheduled overall at the cost of one fewer TPL mission and one fewer WPS mission scheduled exists.

Table 6. Percent of Requested Missions Scheduled Given Resources Requested $\leq$ 4992

‘	Max Mission	Priority	Priority Relaxation
All Missions	84.0%	83.7%	83.7%
TPL	98.4%	100.0%	100.0%
WPS	98.1%	99.1%	99.1%

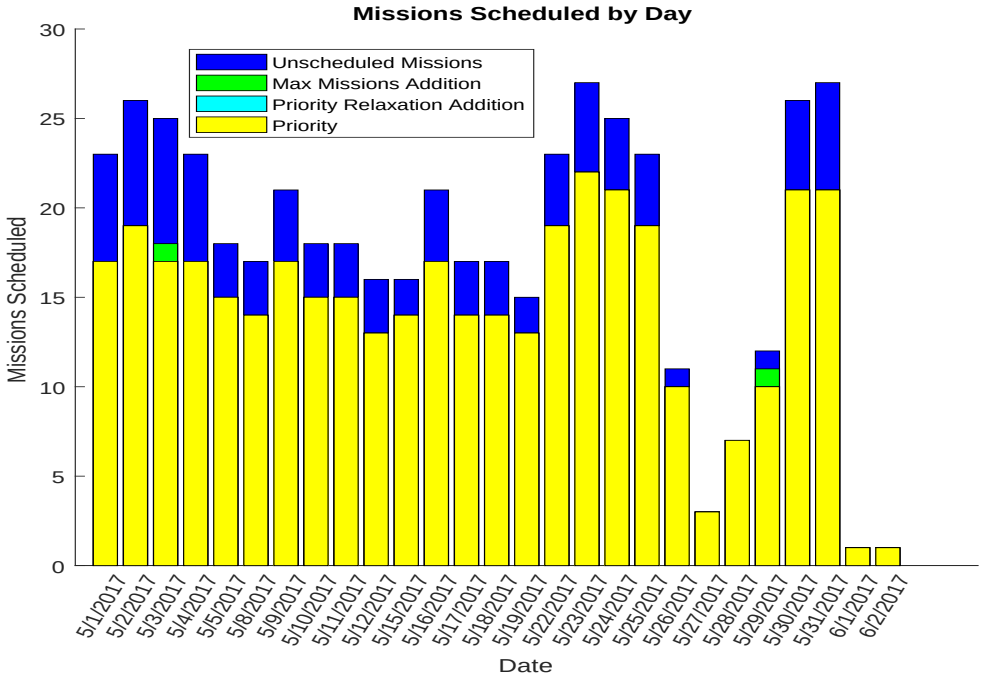

Figure 4. *Missions Scheduled by Day*

Resources Scheduled

When assessing the number of resources scheduled we found that the Max Mission formulation scheduled the statistically significant largest proportion of resources requested among the three formulations. The priority formulations, however, produced total resource scheduling within 2% of the Max Mission method which suggests that the priority based formulations still produce formidable schedules that nearly maximize feasible resource allocations. Table 7 summarizes these results.

Table 7. Percent of Requested Resources Scheduled

	Max Mission	Priority	Priority Relaxation
All Missions	59.7%	58.4%	58.4%

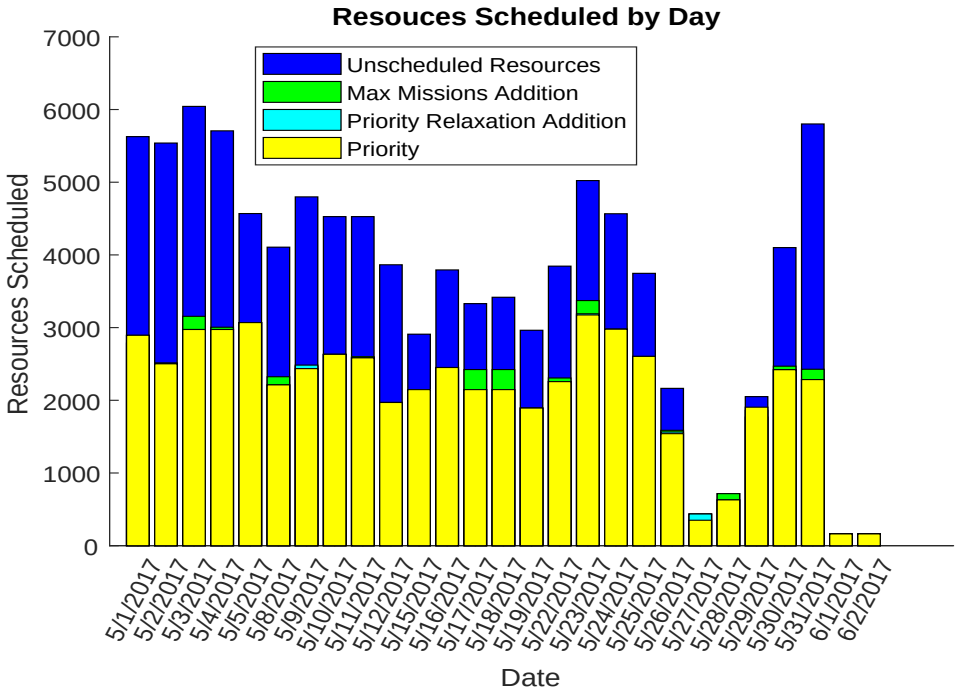

Figure 5. *Resources Scheduled by Day*

We also analyzed the performance of the three formulations under the conditions that the resources requested for the day were less than or equal the total resources

available to be scheduled. The total proportion of resources scheduled increases for all three formulations. Table 8 summarizes these results.

Table 8. Percent of Requested Resources Scheduled Given Requested $\leq$ 4992

	Max Mission	Priority	Priority Relaxation
All Missions	64.2%	62.8%	62.8%

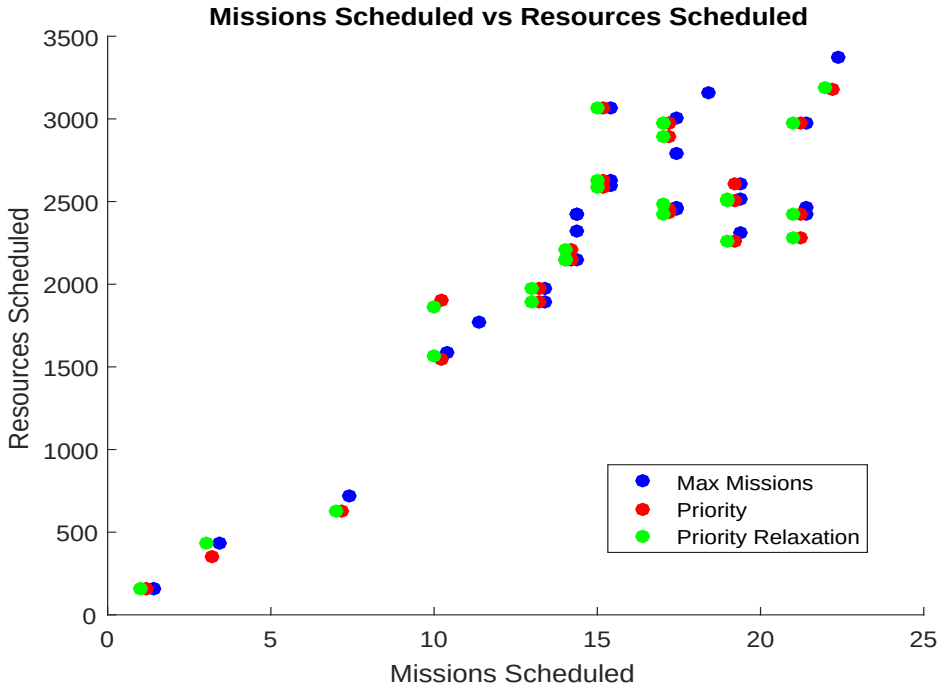

Figure 6. *Missions Scheduled vs Resources Scheduled*

Run Time

We recorded the total run time for the program to schedule a month's worth of missions. We alloted the solver up to four minutes to find the best possible feasible solution. We found that most daily schedules found optimal solutions well within the four minute time limit. As we display in Table 9, only a small minority of the 27 daily schedule solutions used all four minutes to output a solution. We do acknowledge that

in the cases where the solver terminated due to the time limit, we are not guaranteed to have found an optimal solution, just a good feasible solution. We found that all instances of this occur when the resources requested are near or exceed the amount of available resources in the schedule space.

Table 9. Solutions Using Maximum Allotted Time

	Max Mission	Priority	Priority Relaxation
All Missions	2	5	4

When we compared formulations, the Max Mission formulation produced statistically significant faster solutions than both the Priority and Priority Relaxation formulations. These results are consist with our expectations, as we expected the Priority formulation to be the most difficult problem to solve and the Max Mission formulation to be the easiest to solve. Despite their differences, all formulations completed the monthly schedule in less than 32 minutes, which is a significant improvement upon the current practice at Nellis. We summarized the total run time results in Table 10.

Table 10. Total Run Time (mm:ss)

	Max Mission	Priority	Priority Relaxation
All Missions	14:15	31:06	20:02

Branch and Bound Nodes Explored

After running the solver for each daily schedule, we recorded the number of Branch-and-Bound nodes explored. A count of nodes above zero, indicates that the solver could not find an optimal solution using solely LP Relaxation and Cutting Plane methods. We found that on six occasions for the Max Mission formulation, on seven occasions for the Priority formulation, and on eight occasions for the Priority Relaxation formulation the solver did not use Branch-and-Bound to solve the problem. This implies that the majority of the daily schedules use Branch-and-Bound to find

solutions. Branching tree sizes vary depending on problem size and the formulation type. The Max Mission formulation used statistically significant fewer nodes than the Priority formulation but not the Priority Relaxation formulation. The Priority Relaxation formulation uses statistically significant fewer nodes than the Priority formulations. Table 11 summarizes the results for our formulations under our baseline assumptions.

Table 11. Branch-and-Bound Nodes Explored by Daily Schedule

	Max Mission	**Priority**	**Priority Relaxation**
Average	5,544.33	16,255.11	7,622.56
Max	34,099	84,600	46,809
Min	0	0	0

4.4 Adding Resources

In some instances, MARSA (Military Assumes Responsibility for Separation of Aircraft) scheduling allows for multiple missions to occupy the same airspace. We identified the following subranges as candidates for two missions to occupy simultaneously: 61A, 61B, 62A, 65A, 65B, 65C, 65D, 64A, 64B, 64C, 63B, 63H. We modified the RHS of the A matrix to allow for all time blocks associated with the previously mentioned subranges to include two resources. The modified RHS results in 1,152 more resources being added to the problem. With the expectation that more missions with longer durations would get scheduled, we also produced schedules using a revised priority structure, TPL ascending, WPS affiliation, and duration descending.

The Max Mission method yielded 16 more total missions, one more TPL mission and the same number of WPS missions as the schedule built with the 4,992 resources. Under the baseline priority structure with added resources, the priority formulations scheduled more total missions and equivalent amounts of TPL and WPS missions than the schedules built with the baseline number of resources. The Priority formulation

scheduled 19 more missions and the Priority Relaxation formulation scheduled 17 more missions. Figure 7 displays these differences graphically.

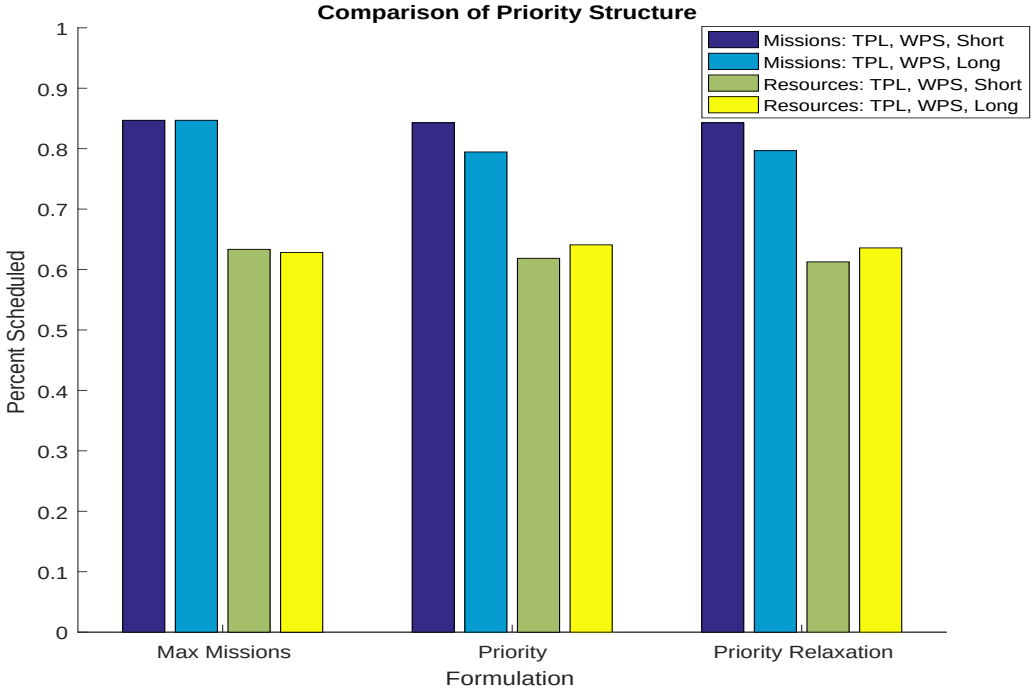

Figure 7. *Comparison of Priority Structures*

Under the revised priority structure, both priority formulations produced fewer total missions but more total resources scheduled. We attribute this difference to the scheduling of more long duration missions that occupy large portions of the feasible schedule space.

Table 12. Percent of Requested Resources Scheduled with Special RHS

	Max Mission	**Priority**	**Priority Relaxation**
Priority: TPL, WPS, Shortest Duration	63.3%	61.9%	61.3%
Priority: TPL, WPS, Longest Duration	62.8%	64.1%	63.6%

When we assess the total run time to find feasible solutions, we found the revised priority structure used more computational time for all formulation types. All formulations find feasible solutions in under 40 minutes which is still significantly faster

than the current practice at Nellis. Table 13 summarizes these results.

Table 13. Total Run Time (mm:ss) with Special RHS

	Max Mission	Priority	Priority Relaxation
Priority: TPL, WPS, Shortest Duration	14:14	24:57	19:45
Priority: TPL, WPS, Longest Duration	18:56	39:10	26:51

Once again, we found that only a small minority of the 27 daily schedule solutions used all four minutes to output a solution. Table 14 summarizes these results. We found that all instances of this occur when the resources requested are greater than or equal to 4,000.

Table 14. Solutions Using Maximum Allotted Time

	Max Mission	Priority	Priority Relaxation
Priority: TPL, WPS, Shortest Duration	2	4	4
Priority: TPL, WPS, Longest Duration	3	5	2

For all formulation types, the baseline priority structure used a larger branching tree on average than the revised priority structure. We believe this may be due to the baseline priority structure having more candidate optimal solutions than the revised priority structure, since shorter duration missions consume fewer resources and make larger objective function contributions under the baseline priority structure. Table 15 summarizes the differences.

Table 15. Difference in Branch-and-Bound Nodes Explored: (Priority: TPL, WPS, Shortest Duration) - (Priority: TPL, WPS, Longest Duration)

	Max Mission	Priority	Priority Relaxation
Average	2,298.30	3,140.85	2,546.63
Max	9,430	23,827	35,371
Min	0	0	0

4.5 Rolling Horizon Scheduling

We explored another scenario in which unscheduled missions from the previous days roll over into the requests for the following day's schedule. We formulated the problem to identify the maximum number of missions that could be scheduled under this condition. When we maintain our assumption that no more than one mission can be active in a subrange at any given time, we found that we could schedule 418 total missions across the 27 day planning horizon. Figure 8 displays these results, using the number of candidate missions and missions scheduled by day. When we add the same resources to the problem that we did in the previous section, we found that 435 missions could be scheduled. Figure 9 displays these results graphically, in a similar manner as described above.

Table 16. Rolling Horizon Scheduling Performance

	Standard RHS	**Special RHS**
Percent Missions Scheduled	87.6%	91.2%
Percent Resources Scheduled	70.2%	73.4%
Run Time (mm:ss)	13:57	16:32

The current priority based formulations do not support rolling horizon scheduling as large numbers of candidate missions produce large objective function coefficient values that stress current computational capabilities. Should a decision maker require priority based rolling horizon scheduling, a revised objective function structure will be necessary.

4.6 Conclusion

In this chapter, we reported and analyzed the results of schedules built using the test data set. We found that priority formulations achieve high quality solutions that require few tradeoffs from maximized mission solutions. Strategically adding resources

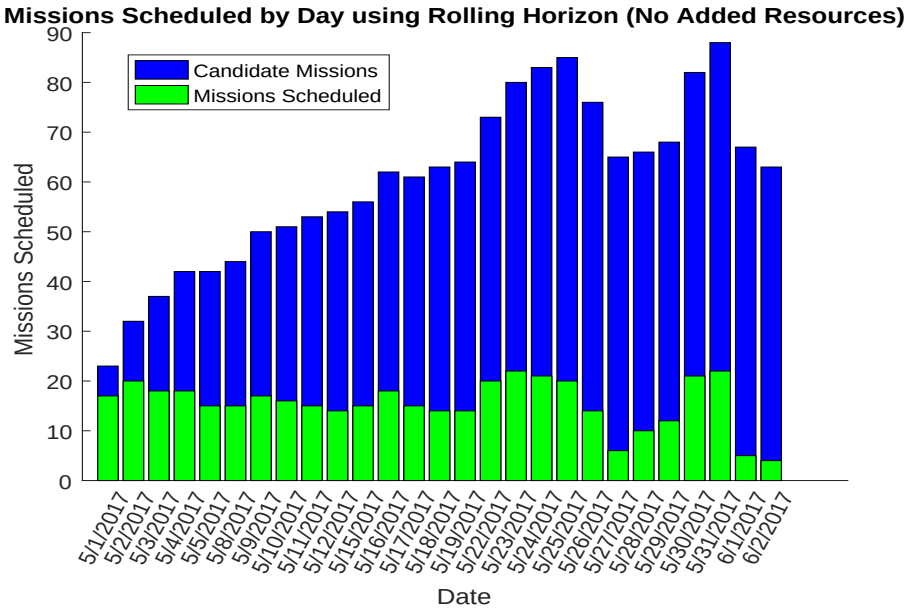

Figure 8. *Rolling Horizon Scheduling by Day with No Added Resources*

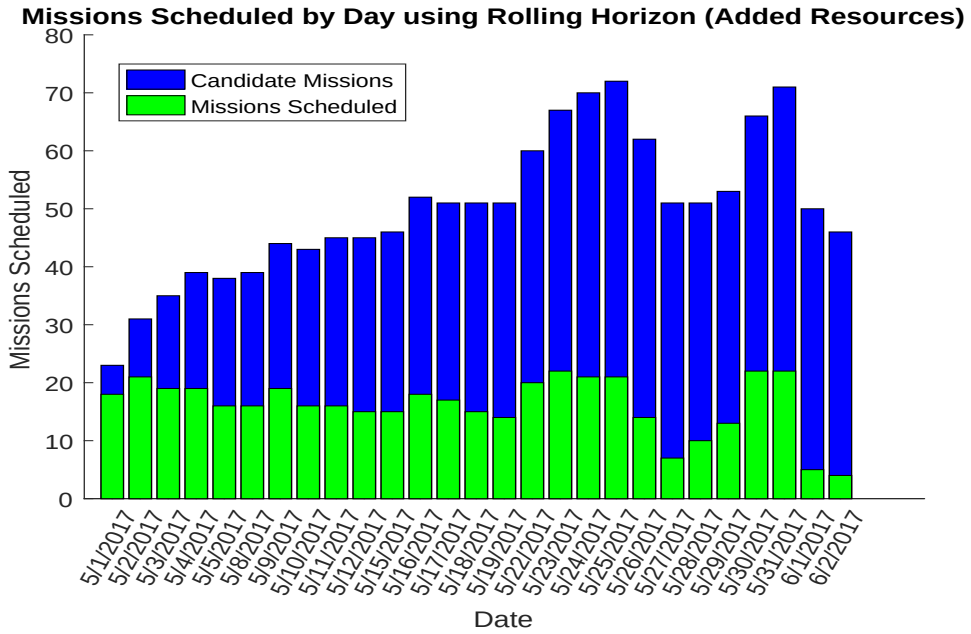

Figure 9. *Rolling Horizon Scheduling by Day with Added Resources*

to the problem has the potential to further increase the number of missions that get scheduled. Though rolling horizon scheduling cannot currently support priority based scheduling, it has the potential to further increase the total number of missions scheduled each day.

V. Conclusions and Future Research

5.1 Conclusion

This study has demonstrated that an IP approach used to schedule Nellis mission requests to resources on the NTTR is feasible and has utility to the scheduling organization. We identified other optimization tools, such as heuristics, that could be used to to solve the problem but found that an IP can find optimal solutions for the current problem sizes and degrees of complexity. We formulated the problem under three different sets of objective function coefficients and analyzed the differences in the schedules they produced from a test data set used to build schedules in May 2017. We found trade offs between the solutions each formulation produced that could be presented to a decision maker.

Under our current understanding of decision maker priorities, we used two priority based objective function coefficient sets to schedule 100% of requested TPL and 94.7% of requested WPS missions. Overall, we maintained scheduling rates above 80% for all missions under the scarcest available resource conditions. We found that solution quality could improve with the addition of resources to the problem. The addition of strategically placed resources to the problem increased the overall scheduling rate to as high as 84.7%. Finally, the rolling over of unscheduled missions has the potential to further increase the overall scheduling rate. When we rolled over unscheduled missions into the following days' requests and added resources to the problem, our overall scheduling rate increased to 91.2%. At this time priority based objective function formulations cannot be combined with rolling horizon scheduling due to computational limitations.

5.2 Recommendations

Based on our analysis and results from testing done on the May 2017 data set, we recommend that 57 OSS/OSOS employ one of the priority based formulations to find fast and good feasible starting solutions to their monthly schedule build. We acknowledge that we could not capture all of the inputs and rationalizations that go into producing schedules, but believe the tool can help expedite the scheduling process and produce efficient NTTR resource allocations. To further improve the tool, we recommend that the existing request system be revised to standardize the format of request data inputs. The output of the tool is only as good as the quality of the input. Standardizing the inputs will help eliminate any discrepancies that cannot currently be detected by the tool and ensure that the tool outputs the highest quality solutions possible.

5.3 Future Research

Since this study is the first effort to implement automation into the NTTR scheduling problem, we did not have the luxury of benefiting from past data and/or work on this problem. We acknowledge that we were not able to include as many aspects to the problem as we had hoped to. More complexity can be added to the problem to account for the scheduling of missions deconflicted by altitude. The availability of mission altitude data is currently sparse but could be obtained through coordination with the requesting units. Additionally, the problem could benefit from more investigation into decision maker goals. The current objective function formulations capture the relative importance of scheduling a mission but not the penalty of failing to schedule a mission or scheduling a mission outside of its requested time range. We believe solutions to the problem could benefit from these additions, so it is our hope that future researchers will continue to build upon our work.

Bibliography

1. J. S. Antes, "An improved scheduling algorithm for eglin afb test ranges," tech. rep., DTIC Document, 1992.

2. P. A. McDaniel, "Estimating test range capacity," tech. rep., DTIC Document, 1993.

3. M. Liljenstolpe, "Single-pass serial scheduling heuristic for eglin afb range services division schedule," tech. rep., DTIC Document, 2009.

4. L. M. Hassel, "Investigation of a zero-one integer programming approach to automating the scheduling process at the usaf test pilot school," tech. rep., DTIC Document, 1991.

5. G. G. Foster, "Automating the weekly flight scheduling process at the usaf test pilot school," tech. rep., DTIC Document, 1992.

6. M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 3rd ed., 2008.

7. R. A. Walker and S. Chaudhuri, "Introduction to the scheduling problem," *IEEE Design & Test of Computers*, vol. 12, no. 2, pp. 60–69, 1995.

8. S. H. Zanakis, J. R. Evans, and A. A. Vazacopoulos, "Heuristic methods and applications: a categorized survey," *European Journal of Operational Research*, vol. 43, no. 1, pp. 88–110, 1989.

9. F. Glover, "Tabu searchpart i," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

10. A. Sarkheyli, A. Bagheri, B. Ghorbani-Vaghei, and R. Askari-Moghadam, "Using an effective tabu search in interactive resources scheduling problem for leo satellites missions," *Aerospace Science and Technology*, vol. 29, no. 1, pp. 287–295, 2013.

11. T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

12. U. Erdemir, "Optimizing flight schedules by an automated decision support system," tech. rep., DTIC Document, 2014.

13. W. L. Winston and J. B. Goldberg, *Operations Research: Applications and Algorithms,Fourth Edition*. Curt Hinrichs, 2004.

14. L. Wolsey and G. Rinaldi, "Integer programming and combinatorial optimization," 1993.

15. R. Impagliazzo, S. Lovett, R. Paturi, and S. Schneider, "0-1 integer linear programming with a linear number of constraints," *arXiv preprint arXiv:1401.5512*, 2014.

16. R. E. Bixby, "A brief history of linear and mixed-integer programming computation," *Documenta Mathematica*, pp. 107–121, 2012.

17. M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*. John Wiley & Sons, 2011.

18. M. L. Fisher, "The lagrangian relaxation method for solving integer programming problems," *Management science*, vol. 27, no. 1, pp. 1–18, 1981.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
22-03–2018	Master's Thesis	Sept 2017 — Mar 2018

4. TITLE AND SUBTITLE

SCHEDULING FOR THE NEVADA TEST AND TRAINING RANGE

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Miguel J. Macias, 1st Lt, U.S. Air Force

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-MS-18-M-138

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

57th Operations Support Squadron
Nellis AFB, NV 89191
Comm: 702-652-3377
Email: rolf.tellefsen@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

57 OSS

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Presently, the 57th Wing Scheduler at Nellis AFB schedules daily mission requests to the Nevada Test and Training Range (NTTR) airspace manually. The process is time consuming and may lead to suboptimal range resource allocations. The goal of this study is to provide the scheduler with an automated scheduling approach that will improve range scheduling efficiency. The tool developed uses range request data from units at Nellis AFB to produce daily mission schedules for a month long scheduling horizon with Microsoft VBA code and a commercial Integer Program (IP) solver. Under our current understanding of scheduler priorities, we formulate the problem with three different sets of objective function coefficients to maximize the utility of the schedules. We then analyze the differences in the schedules produced from a test data set comprised of requests from May 2017. Based on our analysis and results from testing, we recommend that the 57th Wing Scheduler employ one of our priority-based formulations to find fast and good feasible starting solutions to their monthly schedule build.

15. SUBJECT TERMS

Optimization, Integer Programming, Range Scheduling

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. J. Weir, AFIT/ENS
U	U	U	U	56	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x4523; Jeffrey.Weir@afit.edu