NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

OPTIMIZING UNMANNED AIRCRAFT SYSTEM SCHEDULING

by

John L. Pearson

June 2008

Thesis Advisor:	W. Matthew Carlyle
Second Reader:	Sergio Posadas

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2008	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Optimizing Unmanned Aircraft System Scheduling		**5. FUNDING NUMBERS**	
6. AUTHOR(S) John L. Pearson			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

Unmanned Aircraft Systems (UASs) are critical for future combat effectiveness. Military planners from all branches of the Department of Defense now recognize the value that real time intelligence and surveillance from UASs provides the battlefield commander. The Operations Analysis Division of the Marine Corps Combat Development Command is currently conducting an Overarching Unmanned Aircraft Systems study to determine future force requirements. Current analysis is conducted through the use of the Assignment Scheduling Capability for Unmanned Air Vehicles (ASC-U) and several specially designed heuristics. The Unmanned Aircraft System Scheduling Tool (UAS-ST) combines these capabilities into one model and addresses several issues associated with ASC-U. UAS-ST allows the user to control all aspects of the UAS, define a scenario, and then generates a flight schedule over a known time horizon based on those inputs. All missions are assigned a user defined value and the total schedule value is reported. The user can then quickly change a parameter of the UAS, re-solve the model, and see the impact their proposed change has on the overall value of the schedule attained. Therefore, UAS-ST is a tool for analyzing the value of future changes in UAS structure.

14. SUBJECT TERMS Unmanned Aircraft, Optimization, Linear Program, UAS			**15. NUMBER OF PAGES** 95
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

OPTIMIZING UNMANNED AIRCRAFT SYSTEM SCHEDULING

John L. Pearson
Captain, United States Marine Corps
B.S., The Citadel, 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2008

Author: John L. Pearson

Approved by: W. Matthew Carlyle
Thesis Advisor

Sergio Posadas
Second Reader

James N. Eagle
Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Unmanned Aircraft Systems (UASs) are critical for future combat effectiveness. Military planners from all branches of the Department of Defense now recognize the value that real time intelligence and surveillance from UASs provides the battlefield commander. The Operations Analysis Division of the Marine Corps Combat Development Command is currently conducting an Overarching Unmanned Aircraft Systems study to determine future force requirements. Current analysis is conducted through the use of the Assignment Scheduling Capability for Unmanned Air Vehicles (ASC-U) and several specially designed heuristics. The Unmanned Aircraft System Scheduling Tool (UAS-ST) combines these capabilities into one model and addresses several issues associated with ASC-U. UAS-ST allows the user to control all aspects of the UAS, define a scenario, and then generates a flight schedule over a known time horizon based on those inputs. All missions are assigned a user defined value and the total schedule value is reported. The user can then quickly change a parameter of the UAS, re-solve the model, and see the impact their proposed change has on the overall value of the schedule attained. Therefore, UAS-ST is a tool for analyzing the value of future changes in UAS structure.

THIS PAGE INTENTIONALLY LEFT BLANK

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Unmanned Aircraft Systems (UASs) procurement is vital to the United States Marine Corps' (USMC) combat effectiveness in the near future. UASs are used for collecting intelligence, surveillance, and targeting information. They accomplish these missions at a much lower overall risk than conventional, manned aircraft. Military planners from all branches of the Department of Defense (DoD) recognize the value that real time intelligence and surveillance from UASs provides the battlefield commander. In an ongoing operation, mission requests from units in theater typically far exceed the capacity of available UAS assets. Demand for UAS missions is increasing as the capability of these platforms expands. Individual branches of DoD are scrambling to acquire the UASs needed to support the requirements of currently deployed units in combat operations. As a result, interoperability and compatibility is a major concern with today's current family of UASs.

Due to operational requirements in Iraq and Afghanistan, DoD is focusing procurement strategy away from force transformation and future generation weapons to more immediate concerns. Acquisition of UASs needed to meet operational requirements is the agency's highest priority. As a result, large procurement budgets exist to fill the supply shortage and meet future operational requirements. Currently DoD is evaluating a large number of alternative UAS programs. This selection process requires an objective analysis of each alternative. The model in this thesis allows the USMC to accomplish their analysis by clearly depicting the impact of changes in system capabilities on a daily flight schedule. Application of the model is not limited to UASs. The model is capable of analyzing any asset which must be scheduled in response to user demands. Therefore, this model has future application in a multitude of other programs.

The current model in use for the Marine Corps Combat Development Command (MCCDC) Overarching Unmanned Aircraft Systems (OUAS) study is the Assignment Scheduling Capability for UAVs (ASC-U) model. The design agency for ASC-U is the U.S. Army Training and Doctrine Command Analysis Center (TRAC). ASC-U is

designed to support the development of an effective UAS force structure. It is a spreadsheet-based decision support tool primarily for allocation and scheduling of assets. ASC-U addresses complexities in military operations and scheduling of multiple moving platforms. ASC-U accepts user parameters that define a scenario then seeks to provide a feasible schedule for available UASs. ASC-U is the first model that enables analysts to address this scheduling problem effectively. ASC-U combines both optimization and simulation to produce a tool with unique capabilities.

This thesis develops an integer linear programming model, UAS-ST, for scheduling UASs. UAS-ST allows the user to define all elements, both operational and performance, of the UAS via an Excel spreadsheet. A schedule generator written in visual basic for this application then takes these elements and generates a user-defined number of individual schedules. This schedule generation is done for every UAV (UAV) that is included in the scenario. Once the predefined number of schedules is generated, UAS-ST creates data files for a General Algebraic Modeling System (GAMS) model. GAMS, through the use of CPLEX, finds a near optimal combination of individual schedules to produce a complete schedule for a user designated time period.

The initial test scenario replicates a generic scenario given in the OUAS study. This replication provides the simplest and most direct comparison of results with ASC-U. The scenario time period is twenty four hours divided into ninety six intervals of fifteen minutes each. This scenario does not refer to a specific country, but is designed solely to provide the framework for determining the value of changes in UAS structure on a given set of mission requests. UAS force structure for the initial scenario represents a small but realistic composition of systems. Tier I is comprised of three separate units with four UAVs and one Ground Control Station (GCS) per unit. Tier II is comprised of three separate units with one UAV and one GCS per unit. Tier III is comprised of a single unit with two UAVs and two GCSs.

UAS-ST is significantly different from ASC-U. ASC-U utilizes an Excel spreadsheet and pulls the data into an Access database. UAS-ST applies an Excel spreadsheet, which then uses Visual Basic to transfer the data to GAMS. GAMS then applies the CPLEX solver to quickly optimize the schedule for all mission requests.

Features that require supplemental heuristics in ASC-U are incorporated directly into UAS-ST, eliminating the need for further processing. This produces a much quicker and efficient analysis of alternatives. CPLEX is highly efficient in preprocessing feasible solutions and reduces the run time to a matter of minutes. Overall, UAS-ST provides an efficient update to the model presented in ASC-U.

UASs are critical to our nation's and the USMC's future military combat effectiveness. All branches of DoD recognize the need to develop an integrated network of UASs. To answer this need, the Marine Corps is developing the concept of the UAS Family of Systems (FoS). The FoS calls for a three tier structure of UASs with overlapping capabilities. Currently, the MCCDC Operation Analysis Division is conducting an OUAS study. This thesis is a direct contribution to that study, and UAS-ST, provides a planning tool for development of future UAS structure. UAS-ST allows the user to control all aspects of the UAS, define a scenario, and then generates a flight schedule over a known time horizon based on those inputs. All missions are assigned a user-defined value and the total schedule value is reported. The user can then quickly change an operational or performance parameter of the UAS, re-solve the model, and see the impact on the overall value of the schedule. UAS-ST is a tool for analyzing the value of future changes in UAS structure.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PURPOSE AND OVERVIEW

Unmanned Aircraft System (UAS) procurement is vital to the United States Marine Corp's (USMC) combat effectiveness in the near future. UASs are used for collection of intelligence, surveillance, and targeting information. They accomplish these missions at a much lower risk than conventional, manned aircraft. Military planners from all branches of the Department of Defense (DoD) recognize the value that real time intelligence and surveillance from UASs provides the battlefield commander. In an ongoing operation, mission requests from units in theater typically far exceed the capacity of available UAS assets. Future demand for UAS missions will increase as the capability of these versatile platforms expands. Individual branches of DoD are scrambling to acquire the UASs needed to support the requirements of currently deployed units in combat operations. [IHT, 2008] As a result, interoperability and compatibility is a major concern with today's current family of UASs.

To address these issues in future systems, the USMC is developing the UAS Family of Systems (FoS). Each system consists of an Unmanned Air Vehicle (UAV), Ground Control Station (GCS), Launch and Recovery Station (LRS) and a combination of various sensors and payload components. The components of the FoS have several complementary capabilities which overlap in certain mission areas. The intent is to create a mix of several UASs able to support various units of different sizes and levels of operation.

To assess the effectiveness of a combination of UASs, quantitative models must be applied to provide a reasonably accurate measurement of capability and some guidance for efficient employment. To this end, this thesis develops a UAS planning and decision support tool that takes as input a planning horizon, a fleet of UAVs and their individual operating limits, a list of available payloads and a list of mission requests. The model then provides as output an operational schedule for each individual vehicle indicating the missions to be accomplished in a specified time horizon and the payloads

required. This schedule provides analysts the ability to quickly determine the impact of changes in any system parameter on the overall value of missions accomplished.

B. BACKGROUND

1. Problem Statement

Due to operational requirements in Iraq and Afghanistan, DoD is focusing procurement strategy away from force transformation and future generation weapons to more immediate concerns. Acquisition of the UASs needed to meet operational requirements is the agency's highest priority. [IHT, 2008] As a result, large procurement budgets exist to fill the supply shortage and meet future operational requirements. Currently DoD is evaluating a large number of alternative UAS programs. The selection process requires an objective analysis of each alternative. This thesis develops a model to help the USMC with this analysis. The model in this thesis allows the USMC to accomplish their analysis by clearly depicting the impact of changes in system capabilities on a daily flight schedule. Application of the model is not limited to UASs. The model is capable of analyzing any asset which must be scheduled in response to user demands. Therefore, this model has future application in a multitude of other programs.

In recent comments, Defense Secretary Robert Gates, addresses wasteful or inefficient UAS procurement programs in a speech given at the Air Force's Air University at Maxwell Air Force Base:

> Because people were stuck in old ways of doing business, it's been like pulling teeth. While we've doubled this capability in recent months, it is still not good enough. [IHT, 2008]

The fact that the Defense Secretary is unusually blunt in his criticism of current program development, should serve as proof of the pressure to field future UASs.

2. The Marine Corps Three Tier UAS Family of Systems

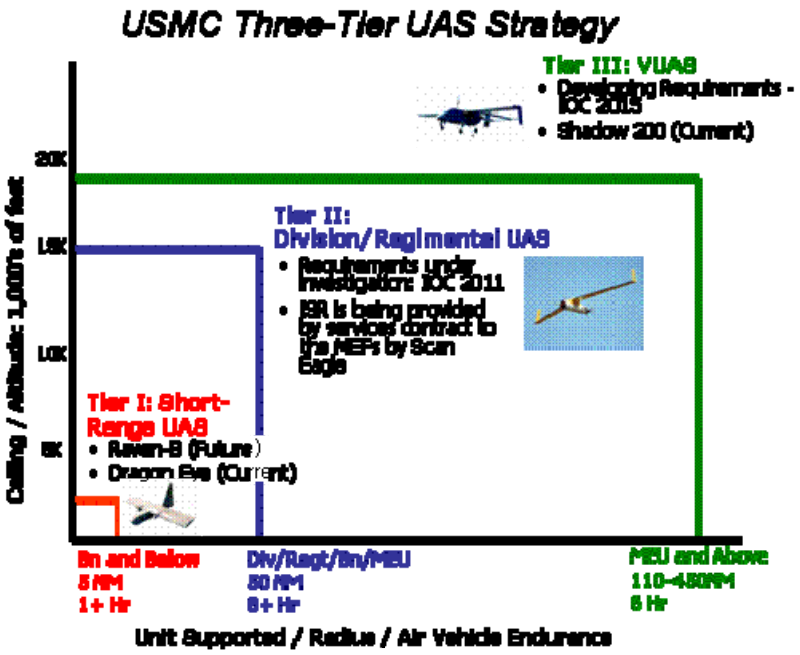

Figure 1. UAS Tier Structure

The FoS divides the various UASs into three separate tiers. The tier assignments provide each level of the Marine Air Ground Task Force (MAGTF) an organic, interoperable, integrated and tailored capability that raises the situational awareness of the combat unit commander through a common communication network. The current operational conditions in Operation Enduring Freedom and Operation Iraqi Freedom demonstrate the importance of UAS operations and the need for commanders at all levels to maintain control of their respective battle space. Therefore, tiers are defined by the area of interest and operational level of the supporting unit, such as a company, battalion, or regiment. There is some operational overlap in tier capabilities; this is intentional and should be considered beneficial to mission accomplishment.

The three tiers are described in [MCCDC, 2005] as follows:

- Tier I – Short Range UAS
    - Operationally Supports Battalion, Company, and Platoon
    - Performs Reconnaissance and Surveillance Missions
    - Current System: Dragon Eye, Future: Raven-B
    - Endurance of 1.5 hours, Combat Radius of 5 miles
    - Speed: Less than 20 knots
    - Capacity for Single Payload

- Tier II – Division or Regimental UAS
    - Operationally Supports Division, Marine Expeditionary Unit (MEU), Regiment and Battalion
    - Air vehicle: Persistent, low cost, durable, low observable, easily transported, shipboard compatible, target acquisition capable, heavy fuel engine
    - Current: Scan Eagle Contract
    - Endurance of 15 hrs, Combat Radius of 50 miles
    - Speed: 60 knots
    - Capacity for Two Payloads

- Tier III – UAS
    - Operationally supports Marine Expeditionary Force, Marine Expeditionary Brigade, MEU, Division and Regiment
    - Reconnaissance Surveillance and Target Acquisition, Electro-Optical /Infrared Imagery (EO/IR)
    - Current System: Pioneer, Future Concept - Vertical Takeoff UAV
    - Air vehicle (Proposed; expeditionary, sea based, Vertical Takeoff, large sensor payload, multi-mission capable, weaponization)
    - Sensor payload (Proposed: EO/IR/laser designation, communication relay, Signals Intelligence, Electronic Warfare)
    - Endurance of 8 hours, Combat Radius of 300 miles
    - Speed: 200 knots
    - Capacity for three payloads

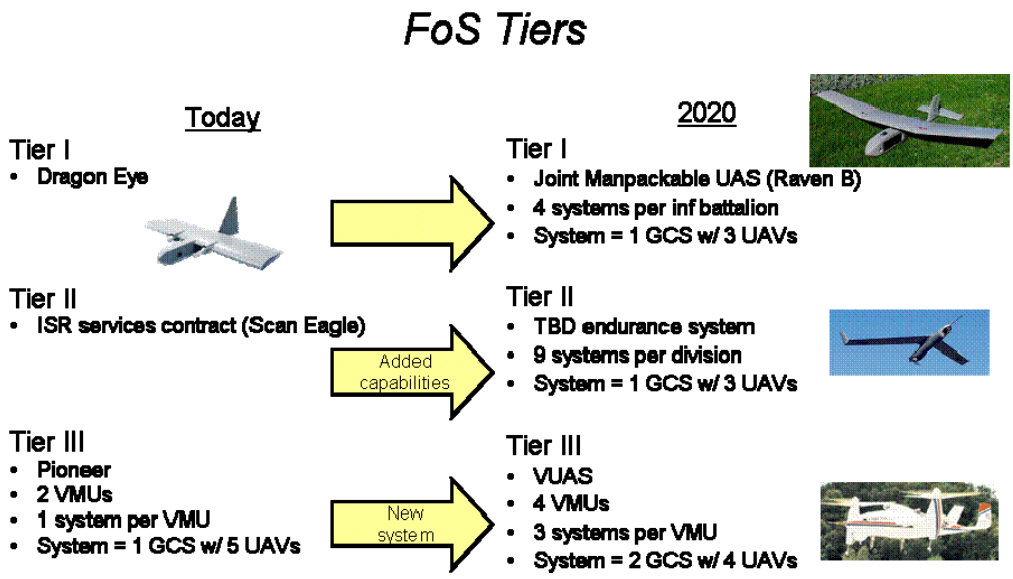

Figure 2. Projected Future FoS Tiers Structure

3. Current UAS Structure

There are several specialized terms associated with a UAS that this thesis uses repeatedly. To avoid any confusion, the following definitions apply:

- **UAV (Unmanned Aerial Vehicle):** A UAV is an unpiloted aircraft. UAVs can be remote controlled or fly an automated route based on pre-programmed information.
- **UAS (Unmanned Aircraft System):** UAS is the current term introduced by DoD and accepted by the Federal Aviation Administration to replace the term UAV. A UAS consists of not only the unmanned aircraft, but also the data link system, the launch and recovery station, and all maintenance and support equipment.
- **GCS (Ground Control Station):** A GCS is a land or sea-based system that allows an operator to control an unmanned aircraft. A GCS may control both "active" and "passive" missions. An active mission requires the operator to monitor the UAV, while a passive mission requires no interaction from the operator.

- **LRS (Launch and Recovery Site):** Used to control an unmanned aircraft during the initial and terminal phases of flight. The LRS can also function as a GCS when not recovering UAVs.
- **Payload:** Future system requirements call for a system of "payloads," each of which provides a custom capability. These payloads will be uniform in size and connectivity to allow for rapid configuration of mission specific profiles.
- **Mission Package:** Consists of the combination of payloads loaded on an individual UAV which determines the mission capability of the UAV. Required payload capacity is a key element of the current study.

4. Current UAS Missions Types

Following is a table of terms associated with current UAS missions. These missions represent general categories and are derived from the Overarching Unmanned Aircraft System (OUAS) study [OUAS, 2007].

Missions	Abbreviation	Description
Reconnaissance, Surveillance, Target Acquisition (RSTA) (EO/IR)	RSTA-EO/IR	ISR asset for routine day and night time operations
Reconnaissance, Surveillance, Target Acquisition (Synthetic aperture radar)	RSTA-SAR	ISR asset for dense vegetation and poor weather conditions
Signals Intelligence	SIGINT	Sensor(s) designed for passive collection of signals
Air Vehicle Communications Link Relay	AV_CLR	Relays information and instructions through one UAV to another
Communications Relay	CR	Relays voice and data between ground points
Strike	STK	Weapons enabled kinetic destruction of a time sensitive target
Electronic Warfare	EW	Active denial of radio frequency

Table 1. General Description of Mission Types

C. SCOPE AND LIMITATIONS

The intent of this thesis is to provide a model for use as an analytical tool in determining the future force structure for UASs. The model allows the user to quickly change both operational and performance parameters for each UAS tier. Once parameters are set, the model quickly generates a near-optimal schedule. This rapid schedule generation allows an analyst to see the impact of changing UAS capabilities on a given daily schedule.

D. THESIS ORGANIZATION

Chapter II provides a discussion of the Assignment Scheduling Capability for UAVs (ASC-U) the current UAS evaluation tool in use by the Operation Analysis Division (OAD), MCCDC. Chapter III describes the optimization model and the stack based enumeration heuristic used to solve it. Chapter IV provides a detailed analysis of the ASC-U results and compares it to the current model. Chapter V is devoted to conclusions and recommendations for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. ASSIGNMENT SCHEDULING CAPABILITY FOR UAS

A. INTRODUCTION

The current model in use for the OUAS study is the ASC-U model. The agency responsible for the design of ASC-U is the U.S. Army Training and Doctrine Command Analysis Center (TRAC). ASC-U is designed to support the development of an effective UAS force structure. This chapter describes the basic structure of the ASC-U model. It begins with a description of the inputs accepted by the model then describes its implementation in the study. The chapter then discusses limitations of ASC-U which are addressed by this thesis. The majority of the information given here is summarized from the OUAS report [OUAS, 2007]. This chapter also includes a review of current literature.

B. BASICS OF ASC-U

1. Model Description

ASC-U supports the development of an effective UAS force structure. It is a spreadsheet-based decision support tool used primarily for allocation and scheduling. ASC-U addresses the complexities in military operations and scheduling of multiple moving platforms. ASC-U accepts user parameters that define a scenario and then seeks to provide a feasible schedule for available UASs. As stated in the ASC-U Analyst Manual:

> ASC-U provides a solution to the following problem: Given a scenario that specifies the number of each type of UAV, initial UAV locations, and UAV performance characteristics, determine the number of missions that can successfully be completed and the schedule for each UAV. The solution must consider GCS locations and capacities, remote viewing terminal requirements, and communication platform footprint and capacities. [Ahner, 2006]

ASC-U is the first model that enables analysts to address this scheduling problem effectively. It combines both optimization and simulation to produce a tool with unique capabilities.

2. ASC-U Implementation

The ASC-U model works by allowing the user to design a scenario which consists of a set of mission requests. A mission request consists of a specific, required UAS capability at a specific geographic location, for a set amount of time. The location of the mission remains fixed once it is assigned. The allocation tool uses the UAS capability data and mission data it is given to create a feasible schedule which will accomplish as many mission requests as possible. ASC-U is deterministic. For a given input, it will always produce the same schedule with the same measures of performance. Specifically, ASC-U uses a deterministic algorithm to optimize over a given finite time horizon to obtain near-optimal UAS mission area assignments.

ASC-U allows the user to define several different input parameters. Mission requirements are the most important parameter. Mission requirements consist of a coordinate location, payload requirement, length of mission and mission priority. Mission priority functions as a selection criterion. A user may establish mission precedence by assigning a higher value to a specific mission type. UAS parameters that may be entered by the user are payload details, GCS attributes, and UAV attributes. GCS attributes consist of coordinate location, control limits, and unit assignment. UAV attributes include speed, operating time, combat radius, launch site, and total time available.

3. Model Capabilities and Limitations

ASC-U is the first attempt to create a specific tool that involves all aspects of the UAS family of systems. As part of its support program, TRAC-Monterey publishes an ASC-U user's manual. For its use in the OUAS study, some settings are different than the manual's recommended settings. Different settings are required due to the small size of the original test scenario chosen. ASC-U is designed to model several thousands of

missions over a long time period. The scenarios used in the OUAS study are much shorter in duration and involve only a few hundred missions.

The mission requirements are the most important input data because they have the most direct effect on the schedule generated. ASC-U attempts to complete as many mission hours as possible. The objective function has a significant drawback. It leads to preferential assignment of missions that are close to the launch and recovery site because they allow for more follow-on missions to be accomplished. The model schedules as many close-in missions as possible and foregoes farther outlying missions. Therefore, mission priority, a specific input, allows the user to set a precedence level. However, in some instances the user is forced to artificially inflate the value of a mission to ensure it is scheduled.

All inputs are usually entered in Excel spreadsheets, and then read into Access. ASC-U can take input from either Excel or Access. The Excel inputs are placed in an Access data base as the model begins its run. The output is also stored in an Access data base. Several tables are produced in the output. The most important table for the OUAS study is mission coverage. Mission coverage is broken down into requesting unit, mission type and UAV type. In ASC-U every possible payload combination is enumerated as a mission package. Mission packages are then assigned to UAVs. Mission package usage is recorded; therefore individual payload usage is not available.

The objective of the first OUAS study is to provide as many hours of UAS support as possible. For ASC-U, the two key factors are the optimization interval and the time horizon for scheduling the UAV. The optimization interval controls how often ASC-U runs its routine to assign missions. The time horizon controls how far forward in time ASC-U will look to assign the UAV to a mission. The optimization interval is crucial because if the interval is set too long then the UAS will remain idle instead of performing another mission. If the interval is set too low the model has difficulties and exhibits odd behavior. An interval of 12 minutes is best for the scenarios used in the OUAS study [OUAS, 2007]. Time horizon is critical because ASC-U will only schedule a UAV once during the given time period. Therefore, if the horizon is set too long, ASC-U can see a high value mission at the end of the period and keep a UAV idle the entire

period waiting for that mission. A UAV can complete another mission and still complete its priority mission. On the other hand, if the time horizon is set too low then ASC-U misses high priority missions because it cannot see far enough out. The final horizon for the OUAS study is 6 hours for Tier II and 4 hours for Tier III.

Because of these shortcomings, several workarounds and new heuristics are used to enable ASC-U to find schedules that exhibit required behaviors. The *SUPER MISSION* ability allows a UAV to fulfill multiple missions if they are within its operating range. Three optional heuristics are also used in the OUAS study. They are implemented in this order:

EARLY RETURN (Go Home vs. Stay) - When a UAV completes an assigned mission, a value of *TRUE* allows the UAV to return to base if it can do so and still make it back on station in time for its next mission. A value of *FALSE* forces the UAV to fly until its operating limit is met. For the OUAS study, this is set to *FALSE*.

SECONDARY AREAS (Go Get More Value From Another Mission) - If a UAV is scheduled for more than one mission in the same location with a time gap in between them, a value of *TRUE* allows the UAV to perform another mission in between as long as it is available at the start of its previously scheduled mission. A value of *FALSE* will force it to remain on station until the start of its next mission. For the OUAS study, this was set to *TRUE*.

APPENDED AREAS (Done, Is There Another Mission?) - If a UAV has completed its assigned missions, a value of *TRUE* allows it to use its remaining time to find another mission. A value of *FALSE* will not allow additional missions. For the OUAS study, this was set to *TRUE*.

C. LITERATURE REVIEW

Many recent studies attempt to shape some aspect of the future design of UAS force structure. An unintended consequence is that most of these studies focus only on some specific technical portion of the overall problem. The Deputy Commandant, Aviation and the Deputy Commandant Combat Development and Integration are

sponsoring the OUAS study. Their intent is to analyze the future USMC UAS force structure to determine how to best meet the needs of the MAGTF. The initial phase of OUAS is being conducted by OAD MCCDC. The model given in this thesis is intended for use in the follow on analysis of OUAS. Therefore, the initial report by OAD is critical to this thesis, as it largely determined the requirements for the model.

Tutton [2003] deals with the optimal placement of a unit's sensing assets. He presents a methodology for finding the most beneficial mix and allocation strategy for an individual unit's sensors for a given threat scenario. Doll [2004] takes the model developed by Tutton [2003] and translates it into a programming language for easier simulation. She refines many of the constraints in the original model to make for a much more realistic simulation. Finally, Zacherl [2006] deals specifically with reactive aircraft scheduling. The thesis develops a model which reviews a current air tasking order and then rapidly reassigns aircraft to new targets as they become available.

Finally a large number of commercial information sources address current topics in UAS development and model optimization. When possible, these sources are used for the sake of currency or accuracy.

THIS PAGE INTENTIONALLY LEFT BLANK

III. OPTIMIZATION MODEL FOR THE SCHEDULING OF UNMANNED AIRCRAFT SYSTEMS

A. INTRODUCTION

This chapter develops the mathematical programming model, Unmanned Aircraft System Scheduling Tool (UAS-ST). UAS-ST applies an optimization based approach to analyze the best mix of future UAS mission capabilities. The main goal of this thesis is to provide a model, which will be applied to the current OUAS study.

The model allows the user to define all elements of the UAS via Excel spreadsheet. The schedule generator then takes these elements and generates a user defined number of individual schedules for every UAV included in the scenario. This schedule generation is done via Excel and Visual Basic. Once the predefined number of schedules is generated, the data is read into a General Algebraic Modeling System (GAMS) program to find a near optimal combination of individual schedules.

B. AN INTEGER PROGRAM TO OPTIMIZE UAS SCHEDULING

The following integer linear program (ILP), UAS-ST, attempts to find the consolidated UAS schedule with the highest overall total value. First the model is presented, then the detailed input required to optimize the objective function is discussed. Once this is complete, instances of UAS-ST are generated to demonstrate its function.

1. Indices

$v \in V$	Tier levels [3]
$h \in H$	Mission types [2] (H= {*passive, active*})
$g \in G$	Ground Control Station (GCS) [~10]
$l \in L$	Launch and Recovery Site (LRS) [~6]
$m \in M$	Missions [~150] (alias *m'*)

$p \in P$	Payload module types [~10]
$t \in T$	Time periods [96]
$s \in S$	UAV employment schedules
$m' \in P_m$	Mission prerequisites: mission *m* cannot be covered in any time period *t* unless at least one mission *m'* in P_m is also covered in time period *t*
$(m, m') \in E$	Pairwise exclusive missions: mission *m* cannot be covered in any time period that mission *m'* is covered. $E \subseteq M \times M$.
$(s, m, t) \in A$	schedule *s* covers mission *m* in time period *t*
$(s, v, t) \in B$	schedule *s* uses UAV in tier level *v* in time period *t*
$(s, l, t) \in K$	schedule *s* uses LRS *l* in time period *t*
$(s, c, t) \in Q$	schedule *s* carries payload of class *c* in time period *t*
$(s, l) \in SL$	schedule *s* uses LRS *l*
$(m, g) \in MG$	mission *m* can be covered by GCS *g*
$(m, h) \in MH$	mission *m* is of mission type *h*
$(g, v) \in GV$	GCS *g* can support a UAS in tier *v*

2. Given Data [units]

val_m	Value per time period of mission *m* (=total value of *m* divided by length of *m*)
r_m	Number of time periods of required coverage for mission *m*
$num\_uavs_{v,t}$	Number of UAVs in tier level *v* available in time period *t*
$num\_pay_{c,t}$	Number of payloads of class *c* available in time period *t*

$gcs\_cap_{g,t}^{h}$ Capacity (in UAVs) of GCS g in time period *t* for missions of type *h*

$lrs\_cap_{l,t}$ Capacity (in UAVs) of LRS *l* in time period *t*

$length_m$ Maximum number of periods mission *m* can be covered

$k_{l,s,t}$ =1 if schedule *s* uses LRS *l* in time period *t*

3. Decision Variables

X_s =1 if schedule *s* flown by some UAS [binary]

$Y_{m,v,t}$ =1 if mission *m* covered by a tier *v* UAS in period *t*

$W_{m,v,g,t}$ =1 if mission *m* supported by GCS *g* for a tier *v* UAS in time period *t* [binary]

D_m Total dwell time on mission *m* [time periods]

$LOAD_{c,v,l}$ Payloads of class *c* for tier *v* sited at LRS *l* [cardinality]

4. Formulation

$$\max \quad \sum_{m} val_m D_m \tag{R0}$$

$$\text{s.t.} \quad \sum_{\substack{s:(s,v,t)\in B \\ (s,l)\in SL}} X_s \le num\_uavs_{l,v,t} \qquad \forall l,v,t \tag{R1}$$

$$\sum_{\substack{s:(s,v,t)\in B \\ (s,c,t)\in Q}} X_s \le LOAD_{c,v,l} \qquad \forall c,v,l,t \tag{R2a}$$

$$\sum_{l} LOAD_{c,v,l} \le num\_pay_{c,v} \qquad \forall c,v \tag{R2b}$$

$$\sum_{\substack{m,v:(m,g)\in MG \\ (m,h)\in MH \\ (g,v)\in GV}} W_{m,v,g,t} \le gcs\_cap^{h}_{g,t} \qquad \forall g,t,h \tag{R3}$$

$$\sum_{s:(s,l,t)\in K} X_s \le lrs\_cap_{l,t} \qquad \forall l,t \tag{R4}$$

$$Y_{m,v,t} \le \sum_{\substack{s:(s,m,t)\in A \\ (s,v,t)\in B}} X_s \qquad \forall m,v,t \tag{R5}$$

$$D_m \le \sum_{v,t} Y_{m,v,t} \qquad \forall m \tag{R6}$$

$$Y_{m,v,t} \le \sum_{m'\in P_m,v'} Y_{m',v',t} \qquad \forall m: P_m \ne \varnothing, \forall v, \forall t \tag{R7a}$$

$$Y_{m,v,t} \le \sum_{\substack{g:(m,g)\in MG \\ (g,v)\in GV}} W_{m,v,g,t} \qquad \forall m: P_m = \varnothing, \forall v, \forall t \tag{R7b}$$

$$\sum_{v} Y_{m,v,t} + \sum_{v} Y_{m',v,t} \le 1 \qquad \forall (m,m') \in E, \forall t \tag{R8}$$

$$D_m \ge r_m \qquad \forall m \tag{R9}$$

$$0 \le D_m \le length_m \qquad \forall m \tag{R10}$$

$$X_s \in \{0,1\} \qquad \forall s \tag{R11}$$

$$Y_{m,v,t} \in \{0,1\} \qquad \forall m,v,t \tag{R12}$$

$$W_{m,v,g,t} \in \{0,1\} \qquad \forall m,v,g,t \tag{R13}$$

5. Discussion

The objective function (R0) calculates the value of all mission time periods covered. Constraints (R1) limit the number of active UAVs in each tier, in each time period, based at each LRS, by the number of UAVS available. Constraints (R2a) limit the number of a specific payload class an tier flown from an LRS in each period to the number of payloads of that class and tier assigned to that LRS, and constraints (R2b) limit the total number of payloads of each class and each tier assigned over all LRSs by the total number available in that class. Constraints (R3) limit the number of missions, by type supported by a GCS in each time period. Constraints (R4) limit the total number of UAVs launching or recovering at a LRS by the capacity of the LRS in a given time period. Constraints (R5) controls whether or not a mission is accomplished in any time period. Constraints (R6) limit the total time spent on a mission to the number of time periods covered. Constraints (R7a) address prerequisite missions such as AV_CLR required for long range Tier III missions, and (R7b) address line-of-sight issues from each GCS for missions that do not have other prerequisites. Constraints (R8) prevent scheduling of mutually exclusive missions by UAS of any tier. Constraints (R9) force "required" missions to be covered for the number of time periods required. Constraints (R10) limit the total dwell time on a mission to between zero and the total amount of time requested for the mission. Constraints (R11-R13) restrict schedule assignment, mission coverage, and GCS assignment to be binary decisions.

6. Route Enumeration and Data Development

UAS-ST requires a significant amount of data processing to produce an optimal schedule; most of this effort is performed by a stack-based enumeration routine that generates all feasible schedules for each tier and LRS in the given scenario. The schedule generator, or simply *generator*, requires two primary data structures for computation of feasible schedules: a stack, PATH(), containing a current feasible list of missions to be accomplished, and an array, ON_STACK(), that indicates whether each mission is currently on the stack of missions. Various data tables are pre-computed to aid in

determining feasibility, such as flight times between airfields and missions, and between pairs of missions, payload requirements by mission, etc.

For each tier, *v*, and LRS, *l*, such that there is at least one UAS of tier *v* located at *l*, the generator starts with an empty mission list PATH(), (representing the trivial action of launching, and then immediately recovering, a UAS of tier *v* from *l*), and builds feasible mission lists exhaustively by adding one mission at a time to the end of the current mission list represented in PATH(). A mission is only added to the end of PATH() if it is within range of a GCS capable of controlling the appropriate tier, if the first period of the mission that can be accomplished (based on the current mission list) allows the UAS to return back to its LRS before running out of operational time available, and if the required list of payloads to accomplish all the missions in PATH() can fit on one UAS of tier *v*. If a mission does not meet these requirements, it is discarded and the next mission from the overall mission list is considered. Other rules for calculating feasible extensions to a current feasible PATH() can be incorporated easily, but these three capture the primary constraints on feasibility. For example, we also check to see if a mission is already on the stack PATH(), and discard those to prevent missions being revisited. However, if the capability to revisit missions in the same schedule is required, this test can be removed. Of course, more feasible schedules will be generated if this is done.

Once a mission is added to PATH(), all missions are considered again for the next empty slot at the end of PATH(). In this manner the generator provides a depth-first exploration of all feasible mission lists. If all missions have been ruled out for a slot, the stack is "popped," the previous slot is then considered again, and the next mission in the list takes the current mission's place; this replacement is repeated until a feasible mission for that slot is found. If no more feasible missions are found for that slot, we "pop" the stack again. When the first slot is finished, we have enumerated all missions from *l* using tier *v* assets.

Every feasible set of missions in PATH() is recorded by incrementing the number of feasible paths found, *s*, and then adding the appropriate elements to the sets

$(s,m,t) \in A$, $(s,v,t) \in B$, $(s,c,t) \in Q$, and $(s,l) \in SL$ to define schedule *s*. See Appendix B for the details of the algorithm.

We set an upper bound on the total number of schedules that can be generated, so as not to create unsolvable ILPs. We also provide a limit on the number of missions enumerated per schedule, and the generator implicitly calculates a limit on the number of schedules to generate for each tier-LRS pair, to allow for the generation of a nonempty list for each such pair regardless of the order in which schedules are generated for those pairs. So, for example, we might set a limit of 120,000 total schedules, and four missions per schedule. The generator calculates how many tier-LRS pairs are feasible, and if, say, there are six such pairs in the scenario, then the generator will generate no more than 20,000 schedules for each pair considered, calculated cumulatively. (Specifically, in this example if the first pair does not generate all 20,000 of its allotted schedules, then the generator can generate up to 40,000 total schedules for the first two pairs combined.) When these enumeration limits are reached for a tier-LRS pair, the stack is cleared out and the generator moves to the next tier-LRS pair.

This procedure can generate tens- or hundreds-of-thousands of feasible schedules. Each such schedule is associated with a decision variable, X_s, in the ILP. The limits on enumeration therefore restrict the schedules available, and, consequently, the resulting ILP will be a restriction of the model that considers *every* feasible schedule. The more schedules that are generated (through increasing the limit on missions per schedule, or the total schedules generated, etc.) the better the final schedule found by the integer program.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. COMPUTATIONAL RESULTS

The OUAS study generic unclassified scenario is replicated with the intent to test the results provided by UAS-ST [OUAS, 2007]. All computations are performed on a Pentium 4 desktop computer at the Naval Postgraduate School with the use of the GAMS solver CPLEX [2008].

A. INITIAL TEST SCENARIO

The initial test scenario replicates a generic scenario given in the OUAS study [OUAS, 2007]. This replication provides the simplest and most direct comparison of results with ASC-U. The scenario time period is twenty four hours divided into ninety six intervals of fifteen minutes each. This scenario does not refer to a specific country, but is designed solely to provide the framework for determining the value of changes in UAS structure on a given set of mission requests. UAS force structure for the initial scenario represents a small but realistic composition of systems. Tier I is comprised of three separate units with four UAVs and one Ground Control Station (GCS) per unit. Tier II is comprised of three separate units with one UAV and one GCS per unit. Tier III is comprised of a single unit with two UAVs and two GCSs.

Table 2 provides a summary of the given UAS force structure for the generic scenario.

Number of:	Tier I	Tier II	Tier III
Systems	12	3	2
UAVs per system	3	3	4
GCSs per system	1	1	2
UAVs controlled by GCS	1	2	2

Table 2. Number of UAS Assets Available in Generic Scenario

The initial test scenario consists of placing units from different UAS Tiers at specific ranges to test the scheduling constraints of the model. The tier III squadron is centered at an airfield in support of a regiment. The squadron consists of two UAVs and

two GCSs at this location. These are the only tier III assets available for our test scenario. The mission areas for the tier III UAS are obtained by plotting points in the cardinal directions at or beyond one hundred and fifty nautical miles, (beyond tier II range).

The three tier II units are deployed in a triangular pattern around the tier III location in direct support of individual battalions. The tier II units are separated from each other by a distance of at least 20 nautical miles. As before, specific mission areas are chosen for each tier II unit.

Tier I units are collocated in the same manner as tier II, but are in direct support at the company level. Each tier I unit consists of four UAVs and a single GCS. Mission areas are selected with a special emphasis on their range. Once the areas are designated, they are used repetitively to generate multiple missions. The use of repetitive mission areas makes it much simpler to detect mission assignments beyond the range of a specific tier. Tier I and II missions are within range of tier III assets.

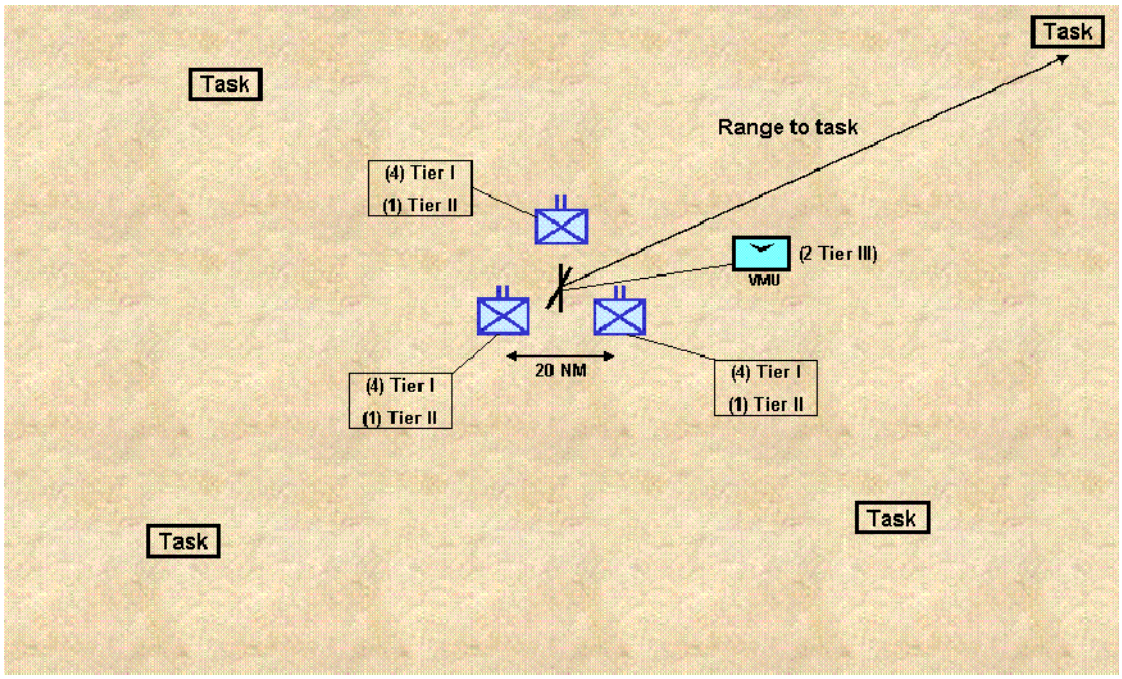

Figure 3. Plot of OUAS Scenario Taken from OUAS

B. UAS-ST INITIAL RESULTS AND ANALYSIS

Data for the initial scenario is entered via a mission request worksheet in Excel. The user enters mission requests for UAS coverage in the following manner:

Mission ID	Location		Mission Class	Start Time	End Time	Total Time Requested	Start Period	Finish Period	Time Periods Requested	Value Per Time Period
	Lat	Long								
m1	N323934	W1144850	AV_CLR	0:00	8:00	8:00	1	32	32	5
m2	N322947	W1144606	ISR	0:30	2:30	2:00	3	10	8	10
m3	N322825	W1144547	ISR	1:00	4:00	3:00	5	16	12	10
m4	N322648	W1144730	ISR	1:30	2:30	1:00	7	10	4	10
m5	N322549	W1144914	ISR	2:00	4:00	2:00	9	16	8	10
m6	N324012	W1142426	ISR	2:30	5:30	3:00	11	22	12	10
m7	N324021	W1142011	ISR	3:00	4:00	1:00	13	16	4	10
m8	N324055	W1142318	ISR	3:30	5:30	2:00	15	22	8	10
m9	N323922	W1142442	ISR	4:00	7:00	3:00	17	28	12	10
m10	N323848	W1142056	ISR	4:30	5:30	1:00	19	22	4	10

Table 3. Input Worksheet for Mission Requests

Waypoints for the mission areas are placed at specific ranges to test the scheduling constraints of the model. These waypoints are then used to generate a table of one hundred and fifty mission requests. The requests are broken down by mission areas. Mission requests one through forty focus on the tier I mission areas. Requests forty one through one hundred and three focuses on tier II and the remaining requests are tier III mission areas. See Appendix A for the complete scenario worksheet used in analysis of UAS-ST. The model constraints allow a higher tier vehicle to perform a lower tier mission if that is the highest value mission available at the time and the vehicle is within range. However, the longer range of each subsequent tier's mission area make it infeasible to schedule a lower tier UAS. The sequential mission numbers enable the user to quickly identify any infeasible mission assignments.

Mission classes are defined according to the mission categories given in the OUAS study. Values for the individual missions are chosen arbitrarily. Selection of the assigned mission values is critical because they have the ability to skew the results by inflating the value of a specific category. In the scenario, active missions are given greater priority than passive. Of the active subset, missions which are unique to a

specific tier are given the highest priority. Missions which are redundant to all tiers are given the lowest priority. Table 4 outlines the missions which each tier can perform and their associated value in the scenario.

	Tier I	Tier II	Tier III
ISR	10	15	20
AV_CLR	5	10	5
CR		10	5
LSR_P		30	
SIGINT			30
EW			40
STK			60
LSR_D			40

Table 4. Assigned Mission Values by Tier

The requested mission time is defined through the start and end times, which are entered by the user. UAS-ST then converts that time into the appropriate number of fifteen minute time periods. Once a UAS is assigned in a given time period it may not be reassigned until the next time period. Times are started at the beginning of the first time period and continue to the end of the final period.

Another element which greatly affects the optimal value achieved is the number of mission payloads available. See Figure 4 for a sample of the payload worksheet. By manipulating the number of payloads of a given type available to a UAS tier, it is possible to test the scheduling constraints and see very quickly if payload availability was a limiting factor. This is highly beneficial in determining future requirements for UAS structure.

Payload ID	Payload Type	Mission Class	Quantity Available	UAS Tier
p1	EO/IR	ISR	9	T1
p2	EO/IR-targeting	ISR_T	3	T2
p3	EO/IR-targeting	ISR_T	2	T3
p4	SAR w/MTI	ISR_SAR	1	T3
p5	SIGINT	SIGINT	1	T2
p6	SIGINT	SIGINT	1	T3
p7	Mine Detection	ISR_MD	1	T3
p8	Pointer	LSR_P	1	T2
p9	Pointer	LSR_P	1	T3
p10	Rangefinder	LSR_RF	1	T3
p11	Designator	LSR_D	1	T3
p12	AV_CLR	AV_CLR	9	T1
p13	AV_CLR	AV_CLR	3	T2
p14	AV_CLR	AV_CLR	1	T3
p15	CR	CR	1	T2
p16	CR	CR	1	T3
p17	Strike	STK	2	T3
p18	EW	EW	2	T3

Figure 4. Example of Payload Worksheet

System capabilities and limitations are drawn directly from the OUAS study. The study seeks to define UAS tier requirements. Therefore, those assumptions are critical to any analysis that is conducted. The tier parameters as given in Figure 5 are used in all testing. Operational time limit and maintenance time are not known at this time. Those columns are for future expansion when the data becomes available.

UAS Tier	Max Endurance (hrs)	Cruise Speed(knots)	Operational Radius(nm)	Payload Capacity	Operational Time Limit(hrs)	Maintenance Time
T1	1.5	17	5	1		
T2	15	60	50	2		
T3	8	200	300	3		

* All Values Taken from Table ES-2 of Overarching UAS Study 21Nov2007

Figure 5. OUAS Study Future Tier Performance Parameters

Once all data for the scenario is entered via Excel spreadsheet, UAS-ST is run from the graphical interface given in Figure 6. The interface, known as the dashboard,

allows the user to specify several key parameters. First the user specifies the start time, the time periods in horizon, and the number of periods per hour. For the mission generation portion, a user specifies the maximum number of schedules that UAS-ST can generate and the maximum number of missions per schedule that a single UAV can perform. These two parameters greatly affect the outcome of the model.

As the number of maximum missions per schedule increases, the number of possible schedules rapidly increases as well. Setting the number of maximum schedules too low prevents a full enumeration of all possible schedules and is a constraint on the model. Setting the number of maximum schedules too high allows a full enumeration of all possible schedules and results in a longer processing time. See Figure 6 for an example. All tested combinations satisfy the requirement for a quick total solution time. Once all parameters are set, selecting the *BUILD* function initiates VBA to build the designated number of flight schedules. After the build is complete, selecting *SOLVE* initiates GAMS to read in the data, apply the selected solver, and generate a near optimal solution. When GAMS reaches a solution, selection of *RESULTS* displays the solution in the format seen in Figure 7.

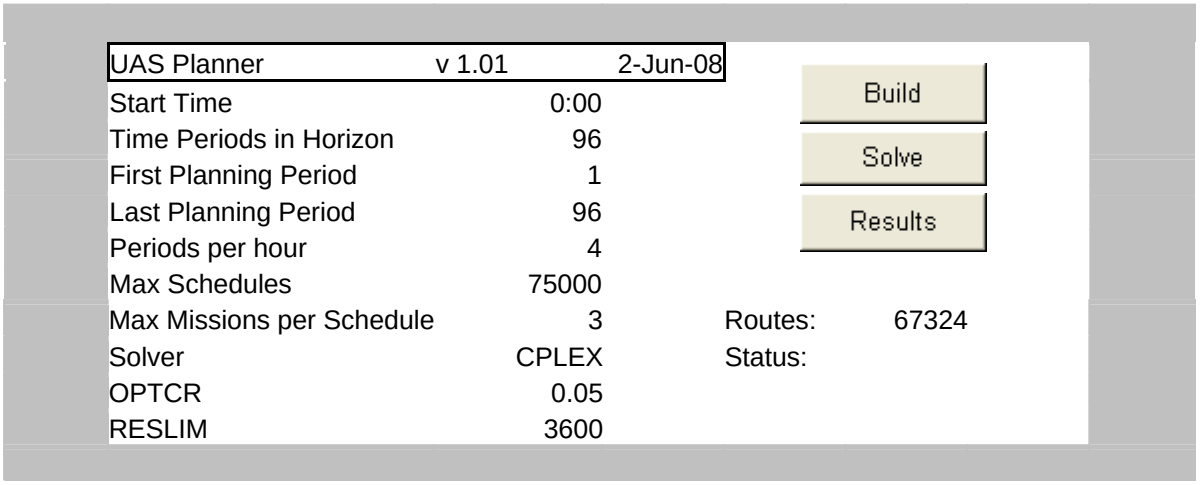

Figure 6. Sample of User Interface (Dashboard)

Mission	Start Time	End Time	Total Time Periods Requested	Total Time Periods Assigned	Value Per Time Period	Total Value Achieved	23340
m1	0:00	8:00	32	0	10	0	
m2	0:30	2:30	8	0	5	0	
m3	1:00	4:00	12	0	5	0	
m4	1:30	2:30	4	0	5	0	
m5	2:00	4:00	8	0	5	0	
m6	2:30	5:30	12	6	5	30	
m7	3:00	4:00	4	1	5	5	
m8	3:30	5:30	8	4	5	20	
m9	4:00	7:00	12	7	5	35	
m10	4:30	5:30	4	0	5	0	
m11	5:00	7:00	8	5	5	25	
m12	5:30	8:30	12	9	5	45	
m13	6:00	7:00	4	0	5	0	
m14	6:30	8:30	8	0	5	0	
m15	7:00	10:00	12	6	5	30	
m16	7:30	8:30	4	0	5	0	
m17	8:00	16:00	32	0	10	0	
m18	8:30	11:30	12	0	5	0	
m19	9:00	10:00	4	0	5	0	
m20	9:30	11:30	8	0	5	0	
m21	10:00	13:00	12	12	5	60	

Figure 7. Partial Display of Optimal UAS-ST Schedule

Figure 7 demonstrates the format in which the solution given by UAS-ST is displayed. All missions are sorted sequentially by their assigned mission number. The total value achieved for each mission is given in the right column. The total value obtained by the entire schedule is given in the top right corner. This format enables the easiest comparison between runs because it quickly lets the user see which missions are scheduled and for what portion of the requested time. If certain missions are not being scheduled, the trend is easy to detect.

C. COMPARING UAS-ST TO ASC-U

ASC-U is the model originally used in the OUAS study. It is highly useful in the analysis of UAS future requirements with several strong attributes. However, there are some areas which require modification to meet the needs of the OUAS study. The intent of this thesis is to develop a user friendly model that addresses those shortcomings and meets the requirements of OAD.

For ease-of-use the individual elements of the UAS are separated into different worksheets. A separate spreadsheet for the UAS tier- specific performance requirements is also included. This spreadsheet allows the user to verify the basic parameters of all UAS tiers in one chart. The user can easily modify a parameter and re-run the model to analyze the effect. The model applies these values to the system constraints when optimizing the schedule.

UAS-ST takes a different approach from ASC-U when addressing the optimization of payloads. In ASC-U all possible combination of payloads are enumerated and then designated as mission packages. UAS-ST treats the payloads as individual units, which allows for tracking of specific payload utilization. OAD indicated this was necessary for determining future requirements. The payload data is further separated by creating a mission class worksheet and a payload worksheet. The mission class worksheet defines all current missions for the UAS. It allows the user to designate a mission as either active or passive and which tier is capable of covering that mission. The payload worksheet allows the user to designate the quantity available.

A major difference between the two models is in the handling of the mission data. In UAS-ST all information is consolidated, both input and output, in the same worksheet. The user builds the missions page in the same manner as a flight schedule and assigns a specific number to an individual mission. UAS-ST sorts all output based on mission number, which makes for easy tracking of a specific mission. ASC-U uses either the mission number or the coordinates of the mission location to sort output. This makes analysis of results extremely time consuming and requires additional sorting. ASC-U requires the use of a heuristic, *SUPER MISSIONS*, to designate a mission as mandatory by assigning it extra value. UAS-ST addresses this in the mission input page by allowing the user to designate any portion of a mission request as required. Setting this requirement serves as an additional constraint on the optimization model.

ASC-U utilizes an Excel spreadsheet and pulls the data into an Access database. UAS-ST applies an Excel spreadsheet, which uses Visual Basic to transfer the data to GAMS. GAMS then applies the CPLEX solver to quickly optimize the schedule for the mission requests it is given. The required features addressed by the supplemental

heuristics in ASC-U are incorporated directly into UAS-ST to eliminate the need for further processing. This allows for a much quicker and efficient analysis of alternatives. CPLEX is highly efficient in preprocessing the feasible solutions and reduces the run time to a matter of minutes. Overall, UAS-ST provides an efficient update to ASC-U.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND NEW OPPORTUNITIES

A. SUMMARY

This thesis develops a scheduling tool, UAS-ST as a replacement for ASC-U. UAS-ST addresses several areas which require additional processing in ASC-U. UAS-ST allows a user to generate a daily flight based on a given set of operational and performance parameters. The true strength of UAS-ST is that a user can change a single parameter or some combination of parameters and quickly rerun the model to see the effect on a flight schedule. UASs are critical to our nation's future combat effectiveness. As a result, all branches of DoD recognize the need to develop an integrated network of UASs. MCCDC OAD is currently conducting an OUAS study with the goal of defining future UAS requirements. This thesis is a direct contribution to that study, and the model developed herein, provides a planning tool for the development of future UAS structure.

B. OPERATIONAL INTRODUCTION

UAS-ST is currently intended for use in the second phase of the OUAS study. It will replace ASC-U and provide another tool for quantitative analysis. UAS-ST provides new capabilities which ASC-U was unable to perform. With the use of the CPLEX solver in GAMS, UAS-ST is able to quickly analyze tens of thousands of schedules for individual UAVs and then choose a near-optimal subset to provide a complete schedule. The user is able to modify all aspects of the UAS structure through a simple spreadsheet interface. Output is provided in the same format, which makes understanding the results and error tracking much simpler. ASC-U requires the use of several heuristics to meet the needs of OUAS. UAS-ST directly addresses these issues without the need for additional heuristics. UAS-ST provides a rapid analysis tool and should be implemented immediately in the ongoing OUAS study.

C. FUTURE DEVELOPMENT

UAS development is expanding rapidly with several new technologies becoming available. Therefore, modeling requirements are expected to change rapidly as well. A possible issue with UAS-ST is the requirement for the GAMS CPLEX solver to produce a quick solution. Due to portability issues, the future development of a non-proprietary heuristic is a valuable topic for future analysis. To improve on UAS-ST, the heuristic needs to modify the enumeration routine used in the model. Rather than a total enumeration of all possible schedules, a new heuristic should initially select higher value routes and not consider low value routes beneath a designated limit. One possible method is a greedy heuristic which selects the highest value subset of individual schedules and eliminate all others from consideration. By initially eliminating low value routes, the overall processing time is greatly reduced. In the long term, analysts in theater may run UAS-ST on their laptops as they plan for actual UAS employment. The ultimate goal is to find a near-optimal solution without the need for specialty software applications.

LIST OF REFERENCES

Ahner, Major Daryl (2006). "ASC-U Users/Analyst Manual" TRAC-Monterey.

Doll, T., (2004) "Optimal Sensor Allocation for a Discrete Event Combat Simulation." MS Thesis in Operations Research, Naval Postgraduate School.

General Algebraic Modeling System (GAMS) 2008. http://www.gams.com (Accessed June 9, 2008).

General Algebraic Modeling System (GAMS) 2008 CPLEX Solver Guide. http://www.gams.com/solvers/cplex.pdf (Accessed June 9, 2008).

International Herald Tribune (August 15, 2007). "United States: Procurement Focuses on Iraq War Needs." http://www.iht.com/articles/2007/08/15/news/15oxan-usmilitary.php. (Accessed June 10, 2008).

Operation Analysis Division, Marine Corps Combat Development Command, Quantico, VA (2007). "Overarching Unmanned Aircraft Systems Study Report."

Tutton, S. (2003). "Optimizing the Allocation of Sensor Assets for the Unit of Action." MS Thesis in Operations Research, Naval Postgraduate School

Zacherl, B. (2006). "Weapon-Target Pairing; Revising an Air Tasking Order in Real Time" MS Thesis in Operations Research, Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A TEST SCENARIO

Tier I Mission Areas (m1-m40)

Tier II Mission Areas (m41-m103)

Tier III Mission Areas (m104-150)

Mission ID	Location		Mission Class	Start Period	Finish Period	Time Periods Requested	Value Per Time Period
	Lat	Long					
m1	N323934	W1144850	AV_CLR	1	32	32	5
m2	N322947	W1144606	ISR	3	10	8	10
m3	N322825	W1144547	ISR	5	16	12	10
m4	N322648	W1144730	ISR	7	10	4	10
m5	N322549	W1144914	ISR	9	16	8	10
m6	N324012	W1142426	ISR	11	22	12	10
m7	N324021	W1142011	ISR	13	16	4	10
m8	N324055	W1142318	ISR	15	22	8	10
m9	N323922	W1142442	ISR	17	28	12	10
m10	N323848	W1142056	ISR	19	22	4	10
m11	N325910	W1142608	ISR	21	28	8	10
m12	N325733	W1142738	ISR	23	34	12	10
m13	N325757	W1142944	ISR	25	28	4	10
m14	N325927	W1142938	ISR	27	34	8	10
m15	N330047	W1142630	ISR	29	40	12	10
m16	N323934	W1144850	ISR	31	34	4	10
m17	N322947	W1144606	AV_CLR	33	64	32	5
m18	N322825	W1144547	ISR	35	46	12	10
m19	N322648	W1144730	ISR	37	40	4	10
m20	N322549	W1144914	ISR	39	46	8	10
m21	N324012	W1142426	ISR	41	52	12	10
m22	N324021	W1142011	ISR	43	46	4	10
m23	N324055	W1142318	ISR	45	52	8	10
m24	N323922	W1142442	ISR	47	58	12	10
m25	N323848	W1142056	ISR	49	52	4	10
m26	N325910	W1142608	ISR	51	58	8	10
m27	N325733	W1142738	ISR	53	64	12	10
m28	N325757	W1142944	ISR	55	58	4	10
m29	N325927	W1142938	ISR	57	64	8	10
m30	N330047	W1142630	ISR	59	70	12	10
m31	N323934	W1144850	ISR	61	64	4	10
m32	N324012	W1142426	ISR	63	70	8	10
m33	N325910	W1142608	AV_CLR	65	96	32	5
m34	N322947	W1144606	ISR	67	70	4	10

m35	N324021	W1142011	ISR	69	76	8	10
m36	N325733	W1142738	ISR	71	82	12	10
m37	N322825	W1144547	ISR	73	76	4	10
m38	N324055	W1142318	ISR	75	82	8	10
m39	N325757	W1142944	ISR	77	88	12	10
m40	N322648	W1144730	ISR	79	82	4	10
m41	N321813	W1145516	ISR	1	4	4	15
m42	N321817	W1143321	ISR	3	18	16	15
m43	N322752	W1150128	ISR	5	36	32	15
m44	N322413	W1140639	AV_CLR	1	48	48	10
m45	N324657	W1135856	CR	1	48	48	10
m46	N325149	W1142341	LSR_P	1	8	8	30
m47	N330310	W1144304	ISR	7	10	4	15
m48	N331121	W1141606	ISR	9	24	16	15
m49	N331641	W1144635	ISR	11	42	32	15
m50	N321813	W1145516	AV_CLR	9	56	48	10
m51	N321817	W1143321	CR	9	56	48	10
m52	N322752	W1150128	LSR_P	17	24	8	30
m53	N322413	W1140639	ISR	13	16	4	15
m54	N324657	W1135856	ISR	15	30	16	15
m55	N325149	W1142341	ISR	17	48	32	15
m56	N330310	W1144304	AV_CLR	17	64	48	10
m57	N331121	W1141606	CR	17	64	48	10
m58	N331641	W1144635	LSR_P	25	32	8	30
m59	N321813	W1145516	ISR	19	22	4	15
m60	N321817	W1143321	ISR	21	36	16	15
m61	N322752	W1150128	ISR	23	54	32	15
m62	N322413	W1140639	AV_CLR	25	72	48	10
m63	N324657	W1135856	CR	25	72	48	10
m64	N325149	W1142341	LSR_P	41	48	8	30
m65	N330310	W1144304	ISR	25	28	4	15
m66	N331121	W1141606	ISR	27	42	16	15
m67	N331641	W1144635	ISR	29	60	32	15
m68	N321813	W1145516	AV_CLR	33	80	48	10
m69	N321817	W1143321	CR	33	80	48	10
m70	N322752	W1150128	LSR_P	57	64	8	30
m71	N322413	W1140639	ISR	31	34	4	15
m72	N324657	W1135856	ISR	33	48	16	15
m73	N325149	W1142341	ISR	35	66	32	15
m74	N330310	W1144304	AV_CLR	41	88	48	10
m75	N331121	W1141606	CR	41	88	48	10
m76	N331641	W1144635	LSR_P	73	80	8	30
m77	N321813	W1145516	ISR	37	40	4	15
m78	N321817	W1143321	ISR	39	54	16	15
m79	N322752	W1150128	ISR	41	72	32	15
m80	N322413	W1140639	AV_CLR	49	96	48	10
m81	N324657	W1135856	CR	49	96	48	10

m82	N325149	W1142341	LSR_P	81	88	8	30
m83	N330310	W1144304	ISR	43	46	4	15
m84	N331121	W1141606	ISR	45	60	16	15
m85	N331641	W1144635	ISR	47	78	32	15
m86	N321813	W1145516	AV_CLR	57	96	40	10
m87	N321817	W1143321	CR	57	96	40	10
m88	N322752	W1150128	LSR_P	89	96	8	30
m89	N322413	W1140639	ISR	49	52	4	15
m90	N324657	W1135856	ISR	51	66	16	15
m91	N325149	W1142341	ISR	53	84	32	15
m92	N330310	W1144304	ISR	55	58	4	15
m93	N331121	W1141606	ISR	57	72	16	15
m94	N331641	W1144635	ISR	59	82	24	15
m95	N321813	W1145516	ISR	61	64	4	15
m96	N321817	W1143321	ISR	63	66	4	15
m97	N322752	W1150128	ISR	65	80	16	15
m98	N322413	W1140639	ISR	67	96	30	15
m99	N324657	W1135856	ISR	69	72	4	15
m100	N325149	W1142341	ISR	71	86	16	15
m101	N330310	W1144304	ISR	73	96	24	15
m102	N331121	W1141606	ISR	75	78	4	15
m103	N331641	W1144635	ISR	77	92	16	15
m104	N312538	W1145510	ISR	1	96	96	20
m105	N334259	W1155906	ISR	1	48	48	20
m106	N333540	W1115204	ISR	49	96	48	20
m107	N312538	W1145510	ISR	1	32	32	20
m108	N334259	W1155906	ISR	33	64	32	20
m109	N333540	W1115204	ISR	65	96	32	20
m110	N312538	W1145510	CR	1	96	96	5
m111	N334259	W1155906	CR	1	96	96	5
m112	N333540	W1115204	CR	1	96	96	5
m113	N312538	W1145510	CR	1	96	96	5
m114	N334259	W1155906	SIGINT	1	96	96	30
m115	N333540	W1115204	SIGINT	1	96	96	30
m116	N312538	W1145510	SIGINT	1	96	96	30
m117	N334259	W1155906	SIGINT	1	96	96	30
m118	N333540	W1115204	EW	1	8	8	40
m119	N312538	W1145510	EW	9	16	8	40
m120	N334259	W1155906	EW	25	32	8	40
m121	N333540	W1115204	EW	33	40	8	40
m122	N312538	W1145510	EW	49	56	8	40
m123	N334259	W1155906	EW	57	64	8	40
m124	N333540	W1115204	EW	73	80	8	40
m125	N312538	W1145510	EW	81	88	8	40
m126	N334259	W1155906	STK	1	16	16	60
m127	N333540	W1115204	STK	17	32	16	60
m128	N312538	W1145510	STK	33	48	16	60

m129	N334259	W1155906	STK	49	64	16	60
m130	N333540	W1115204	STK	65	80	16	60
m131	N312538	W1145510	STK	81	96	16	60
m132	N334259	W1155906	STK	25	40	16	60
m133	N333540	W1115204	STK	73	88	16	60
m134	N312538	W1145510	LSR_D	1	8	8	40
m135	N334259	W1155906	LSR_D	25	32	8	40
m136	N333540	W1115204	LSR_D	49	56	8	40
m137	N312538	W1145510	LSR_D	73	80	8	40
m138	N334259	W1155906	AV_CLR	1	24	24	5
m139	N333540	W1115204	AV_CLR	25	48	24	5
m140	N312538	W1145510	AV_CLR	49	72	24	5
m141	N334259	W1155906	AV_CLR	73	96	24	5
m142	N333540	W1115204	AV_CLR	1	24	24	5
m143	N312538	W1145510	AV_CLR	25	48	24	5
m144	N334259	W1155906	AV_CLR	49	72	24	5
m145	N333540	W1115204	AV_CLR	73	96	24	5
m146	N312538	W1145510	AV_CLR	1	24	24	5
m147	N334259	W1155906	AV_CLR	25	48	24	5
m148	N333540	W1115204	AV_CLR	49	72	24	5
m149	N312538	W1145510	AV_CLR	73	96	24	5
m150	N334259	W1155906	AV_CLR	1	24	24	5

APPENDIX B VBA CODE

```
Sub SolveUAS()
  Dim bSolveResult As Boolean
  bSolveResult = SolveProblem(True)
End Sub
Sub BuildData()
  Dim uasDataFN As Integer
  Dim uasSetsFN As Integer
  Dim uasDynSetsFN As Integer
  Dim logFN As Integer
  Dim rTiers As Range
  Dim rGCSs As Range
  Dim rLRSs As Range
  Dim rUAVs As Range
  Dim rMissions As Range
  Dim rPayloads As Range
  Dim rSchedules As Range
  Dim rClasses As Range
  Dim lNumTiers As Long, lNumGCSs As Long, lNumLRSs As Long, lNumUAVs As
Long
  Dim lNumMissions As Long, lNumPayloads As Long, lNumClasses As Long
  Dim lNumSchedules As Long
  Dim lNumLRSSchedules As Long
  Dim bNextRoute As Boolean

  Dim rValues As Range
  Dim lNumValues As Long
  Dim rDependencies As Range
  Dim lPlanPeriods As Long
  Dim lFirstPeriod As Long
  Dim lLastPeriod As Long
  Dim lPeriodsPerHour As Long

  Dim lCount As Long
  Dim currCell As Range
  Dim iRow As Integer, iCol As Integer
  Dim lMission As Long, lMission2 As Long, lUAV As Long
```

```
        Dim lPeriod As Long, lPeriod1 As Long, lPeriod2 As Long
        Dim lTier As Long, lPayload As Long, lGCS As Long, lLRS As Long
        Dim lClass As Long

        Dim tierNames As StringDictionaryClass
        Dim uavNames As StringDictionaryClass
        Dim gcsNames As StringDictionaryClass
        Dim lrsNames As StringDictionaryClass
        Dim missionNames As StringDictionaryClass
        Dim payloadNames As StringDictionaryClass
        Dim classNames As StringDictionaryClass
        Dim periodNames As StringDictionaryClass
        Dim tValues As TupleDictionaryClass

        uasDataFN = FreeFile()
        Open ThisWorkbook.path & "\uasData.gms" For Output As uasDataFN
        uasSetsFN = FreeFile()
        Open ThisWorkbook.path & "\uasSets.gms" For Output As uasSetsFN
        logFN = FreeFile()
        Open ThisWorkbook.path & "\uasXL.log" For Output As logFN

      ' {STATIC, GROUNDED} SETS
      '   v             Tier levels
      '   h             mission types           {active, passive}
      '   g             Ground control stations
      '   l             Launch and recovery sites
      '   m               Missions                  {missions that ships can be
assigned}
      '   t             Time periods
      '   p             Payload module types
      '   s             UAV employment schedules
      ' ;

        Print #logFN, "Starting output"
        Print #uasSetsFN, " OPTIONS"
        Print #uasSetsFN, "   MIP = " & wsDashboard.Range("SOLVER")
        Print #uasSetsFN, "   OPTCR = " & wsDashboard.Range("OPTCR")
        Print #uasSetsFN, "   RESLIM = " & wsDashboard.Range("RESLIM")
        Print #uasSetsFN, " ;"
```

```
        Print #uasSetsFN, " SETS"
        Set rTiers = wsTiers.Range("A4", wsTiers.Range("A4").End(xlDown))
        Call GAMSWriteSetDef(rTiers, "v", uasSetsFN, "Tier levels")
        Set rUAVs = wsUAV.Range("A4", wsUAV.Range("A4").End(xlDown))
        Print #uasSetsFN, "   h /"
        Print #uasSetsFN, "     h_active"
        Print #uasSetsFN, "     h_passive"
        Print #uasSetsFN, "   /"
        Set rGCSs = wsGCS.Range("A4", wsGCS.Range("A4").End(xlDown))
        Call GAMSWriteSetDef(rGCSs, "g", uasSetsFN, "Ground control stations")
        Set rLRSs = wsLRS.Range("A4", wsLRS.Range("A4").End(xlDown))
        Call  GAMSWriteSetDef(rLRSs,  "l",  uasSetsFN,  "Launch  and  recovery
sites")
        Set            rMissions            =            wsMissions.Range("A4",
wsMissions.Range("A4").End(xlDown))
        Call GAMSWriteSetDef(rMissions, "m", uasSetsFN, "Missions")
        Set            rPayloads            =            wsPayloads.Range("A4",
wsPayloads.Range("A4").End(xlDown))
      '  Call GAMSWriteSetDef(rPayloads, "p", uasSetsFN, "Payloads")
        Set            rClasses            =            wsMClasses.Range("A4",
wsMClasses.Range("A4").End(xlDown))
        Call GAMSWriteSetDef(rClasses, "c", uasSetsFN, "Payload Classes")


        lPeriodsPerHour = CLng(wsDashboard.Range("PERIODS_PER_HOUR"))
        lPlanPeriods = CLng(wsDashboard.Range("HORIZON"))
        lFirstPeriod = CLng(wsDashboard.Range("FIRST_PERIOD"))
        lLastPeriod = CLng(wsDashboard.Range("LAST_PERIOD"))
        If lLastPeriod > lPlanPeriods Then
          MsgBox  "Last  Period  comes  after  maximum  planning  horizon:
truncating."
          lLastPeriod = lPlanPeriods
        End If
        If lFirstPeriod > lLastPeriod Then
          MsgBox  "Empty  planning  horizon:  first  Period  occurs  after  last
Period. Aborting."
          GoTo subExit
        End If
        Print #logFN, "Periods " & lPlanPeriods & "  First: " & lFirstPeriod &
"  Last: " & lLastPeriod
        Print  #uasSetsFN,  "    t  Time  Periods  /t_" & Format(lFirstPeriod) &
"*t_" & Format(lLastPeriod) & "/"
```

```
        Set tierNames = New StringDictionaryClass
        If DefineGroundedSet(rTiers, tierNames) = False Then
          Print #logFN, "Duplicate element in grounded set (tiers). User chose
to cancel: no further output."
          GoTo subExit
        End If
        lNumTiers = tierNames.Count

        For lCount = 1 To lNumTiers
          Print #logFN, Format(lCount) & " : " & tierNames.Item(lCount)
        Next lCount

        Set uavNames = New StringDictionaryClass
        If DefineGroundedSet(rUAVs, uavNames) = False Then
          Print #logFN, "Duplicate element in grounded set (uavs). User chose
to cancel: no further output."
          GoTo subExit
        End If
        lNumUAVs = uavNames.Count

        For lCount = 1 To lNumUAVs
          Print #logFN, Format(lCount) & " : " & uavNames.Item(lCount)
        Next lCount

        Set missionNames = New StringDictionaryClass
        If DefineGroundedSet(rMissions, missionNames) = False Then
          Print #logFN, "Duplicate element in grounded set (missions). User
chose to cancel: no further output."
          GoTo subExit
        End If
        lNumMissions = missionNames.Count

        For lCount = 1 To lNumMissions
          Print #logFN, Format(lCount) & " : " & missionNames.Item(lCount)
        Next lCount

        Set payloadNames = New StringDictionaryClass
        If DefineGroundedSet(rPayloads, payloadNames) = False Then
          Print #logFN, "Duplicate element in grounded set (payloads). User
chose to cancel: no further output."
```

```
          GoTo subExit
        End If
        lNumPayloads = payloadNames.Count

        For lCount = 1 To lNumPayloads
          Print #logFN, Format(lCount) & " : " & payloadNames.Item(lCount)
        Next lCount

        Set classNames = New StringDictionaryClass
        If DefineGroundedSet(rClasses, classNames) = False Then
          Print #logFN, "Duplicate element in grounded set (classes). User
chose to cancel: no further output."
          GoTo subExit
        End If
        lNumClasses = classNames.Count

        For lCount = 1 To lNumPayloads
          Print #logFN, Format(lCount) & " : " & payloadNames.Item(lCount)
        Next lCount

        Set gcsNames = New StringDictionaryClass
        If DefineGroundedSet(rGCSs, gcsNames) = False Then
          Print #logFN, "Duplicate element in grounded set (GCS). User chose to
cancel: no further output."
          GoTo subExit
        End If
        lNumGCSs = gcsNames.Count

        For lCount = 1 To lNumGCSs
          Print #logFN, Format(lCount) & " : " & gcsNames.Item(lCount)
        Next lCount

        Set lrsNames = New StringDictionaryClass
        If DefineGroundedSet(rLRSs, lrsNames) = False Then
          Print #logFN, "Duplicate element in grounded set (LRS). User chose to
cancel: no further output."
          GoTo subExit
        End If
        lNumLRSs = lrsNames.Count
```

```
        For lCount = 1 To lNumLRSs
          Print #logFN, Format(lCount) & " : " & lrsNames.Item(lCount)
        Next lCount

        Set periodNames = New StringDictionaryClass
        periodNames.Size = 3 * lPlanPeriods
        For lCount = 1 To lPlanPeriods
          periodNames.Add "t" & Format(lCount)
        Next lCount

        For lCount = lFirstPeriod To lLastPeriod
          Print #logFN, Format(lCount) & " : " & periodNames.Item(lCount)
        Next lCount

      ' {GIVEN} PARAMETERS
      '   num_uavs(l,v,t) number of uavs available at lrs l in tier v in period
t
      '   num_mods(p,t)   number of payload modules available of ID p in period
t
      '   gcs_cap(g,t,h)  max num UAVs on gcs g in period t on mission type h
      '   lrs_cap(l,t)    max num UAVs at LRS l in period t
      '   length(m)       maximum periods of coverage of mission m
      '   val(m)          value of mission m
      '   k(l,s,t)        indicator if lrs l used by schedule s in period t
      ' ;

        Print #uasDataFN, " PARAMETERS"
      '    num_uavs(l,v,t)    number of uavs available at lrs l in tier v in
period t
        Print #uasDataFN, "   num_uavs(l,v,t) /"
        ReDim lNumUAVinPeriod(1 To lrsNames.Count, 1 To tierNames.Count,
lFirstPeriod To lLastPeriod) As Long
        For lPeriod = lFirstPeriod To lLastPeriod
          For lLRS = 1 To lrsNames.Count
            For lTier = 1 To tierNames.Count
              lNumUAVinPeriod(lLRS, lTier, lPeriod) = 0
            Next lTier
          Next lLRS
          For iRow = 1 To rUAVs.Rows.Count
            If rUAVs(iRow, COL_UAV_FIRST_AVAIL) <= lPeriod And rUAVs(iRow,
COL_UAV_LAST_AVAIL) >= lPeriod Then
```

```
            lLRS = lrsNames.Lookup(rUAVs(iRow, COL_UAV_LRS))
            lTier = tierNames.Lookup(rUAVs(iRow, COL_UAV_TIER))
            If lLRS = 0 Then
              MsgBox "Error: invalid LRS name in UAV list"
              wsUAV.Activate
              rUAVs(iRow, COL_UAV_LRS).Select
              GoTo subExit
            ElseIf lTier = 0 Then
              MsgBox "Error: invalid tier name in UAV list"
              wsUAV.Activate
              rUAVs(iRow, COL_UAV_TIER).Select
              GoTo subExit
            Else
              lNumUAVinPeriod(lLRS, lTier, lPeriod) = lNumUAVinPeriod(lLRS,
lTier, lPeriod) + 1
            End If
          End If
        Next iRow
      Next lPeriod

      For lLRS = 1 To lrsNames.Count
        For lTier = 1 To lNumTiers
          For lPeriod = lFirstPeriod To lLastPeriod
            If lNumUAVinPeriod(lLRS, lTier, lPeriod) >= 1 Then
              Print #uasDataFN, "       l_" & lrsNames.Item(lLRS) & ".v_" &
tierNames.Item(lTier) & ".t_" & lPeriod & " " & _
                 lNumUAVinPeriod(lLRS, lTier, lPeriod)
            End If
          Next lPeriod
        Next lTier
      Next lLRS
      Print #uasDataFN, "   /"

      ReDim numMods(classNames.Count, tierNames.Count) As Long
      For lPayload = 1 To rPayloads.Rows.Count
        iRow = classNames.Lookup(rPayloads(lPayload, COL_PAYLOAD_CLASS))
        iCol = tierNames.Lookup(rPayloads(lPayload, COL_PAYLOAD_TIER))
        numMods(iRow, iCol) = numMods(iRow, iCol) + rPayloads(lPayload,
COL_PAYLOAD_QUANTITY)
      Next lPayload
```

```
      '    num_mods(c,v)    number of payload modules available of class c for
tier v
        Print #uasDataFN, "   num_mods(c,v) /"
        For lClass = 1 To classNames.Count
          For lTier = 1 To tierNames.Count
            Print  #uasDataFN,  "c_"  &  classNames.Item(lClass)  &  ".v_"  &
tierNames.Item(lTier) & " " & numMods(lClass, lTier)
          Next lTier
        Next lClass
        Print #uasDataFN, "   /"

      '   gcs_cap(g,t,h)  max num UAVs on gcs g in period t on mission type h
        Print #uasDataFN, "   gcs_cap(g,t,h) /"
        For lGCS = 1 To lNumGCSs
          For lPeriod = lFirstPeriod To lLastPeriod
            Print #uasDataFN, "     g_" & gcsNames.Item(lGCS) & ".t_" & lPeriod
& ".h_active " & rGCSs(lGCS, COL_GCS_ACTIVE)
            Print #uasDataFN, "     g_" & gcsNames.Item(lGCS) & ".t_" & lPeriod
& ".h_passive " & rGCSs(lGCS, COL_GCS_PASSIVE)
          Next lPeriod
        Next lGCS
        Print #uasDataFN, "   /"

      '   lrs_cap(l,t)    max num UAVs at LRS l in period t
        Print #uasDataFN, "   lrs_cap(l,t) /"
        For lLRS = 1 To lNumLRSs
          For lPeriod = lFirstPeriod To lLastPeriod
            Print #uasDataFN, "     l_" & lrsNames.Item(lLRS) & ".t_" & lPeriod
& " " & rLRSs(lLRS, COL_LRS_ACTIVE)
          Next lPeriod
        Next lLRS
        Print #uasDataFN, "   /"

      '   length(m)       maximum periods of coverage of mission m
        Print #uasDataFN, "   length(m) /"
        For lMission = 1 To lNumMissions
          Print #uasDataFN, "      m_" & missionNames.Item(lMission) & " " &
rMissions(lMission, COL_MISSION_PERIODS)
        Next lMission
        Print #uasDataFN, "   /"
```

```
      '   val(m)          value of mission m
        Print #uasDataFN, "   val(m) /"
        For lMission = 1 To lNumMissions
          Print #uasDataFN, "      m_" & missionNames.Item(lMission) & " " &
rMissions(lMission, COL_MISSION_VALUE)
        Next lMission
        Print #uasDataFN, "   /"


      '   Calculate distances between GCS-Mission pairs, LRS-mission pairs, and
mission-mission pairs
        ReDim dDistGM(lNumGCSs, lNumMissions) As Double
        ReDim dDistLM(lNumLRSs, lNumMissions) As Double
        ReDim dDistMM(lNumMissions, lNumMissions) As Double


        Dim i As Long, j As Long


        For lGCS = 1 To rGCSs.Rows.Count
          i = gcsNames.Lookup(rGCSs(lGCS, COL_GCS_NAME))
          For lMission = 1 To rMissions.Rows.Count
            j = missionNames.Lookup(rMissions(lMission, COL_MISSION_NAME))
            dDistGM(i,   j)   =   SphericalDistance(rGCSs(lGCS,   COL_GCS_LAT),
rGCSs(lGCS, COL_GCS_LON), _
                                                rMissions(lMission,
COL_MISSION_LAT), rMissions(lMission, COL_MISSION_LON))
          Next lMission
        Next lGCS


        For lLRS = 1 To rLRSs.Rows.Count
          i = lrsNames.Lookup(rLRSs(lLRS, COL_LRS_NAME))
          For lMission = 1 To rMissions.Rows.Count
            j = missionNames.Lookup(rMissions(lMission, COL_MISSION_NAME))
            dDistLM(i,   j)   =   SphericalDistance(rLRSs(lLRS,   COL_LRS_LAT),
rLRSs(lLRS, COL_LRS_LON), _
                                                rMissions(lMission,
COL_MISSION_LAT), rMissions(lMission, COL_MISSION_LON))
          Next lMission
        Next lLRS


        For lMission = 1 To rMissions.Rows.Count
          i = missionNames.Lookup(rMissions(lMission, COL_MISSION_NAME))
          For lMission2 = 1 To rMissions.Rows.Count
```

```
            j = missionNames.Lookup(rMissions(lMission2, COL_MISSION_NAME))
            dDistMM(i,      j)      =      SphericalDistance(rMissions(lMission,
COL_MISSION_LAT), rMissions(lMission, COL_MISSION_LON), _
                                                rMissions(lMission2,
COL_MISSION_LAT), rMissions(lMission2, COL_MISSION_LON))
          Next lMission2
        Next lMission


      'Find GCSs within range of each mission.  If none, require one of the
AV_CLR missions
        Dim pFN As Integer, hpFN As Integer
        Dim numPreq As Long
        numPreq = 0
        pFN = FreeFile()
        Open "p.gms" For Output As pFN
        hpFN = FreeFile()
        Open "hp.gms" For Output As hpFN
        Print #pFN, " SET P(m,mp) /"
        Print #hpFN, " SET HP(m) /"
        For lMission = 1 To missionNames.Count
          For lGCS = 1 To gcsNames.Count
            If dDistGM(lGCS, lMission) <= rGCSs(lGCS, COL_GCS_LOS) Then GoTo
nextMissionGCS
          Next lGCS
          numPreq = numPreq + 1
          Print #hpFN, "   m_" & missionNames.Item(lMission)
          For lMission2 = 1 To rMissions.Rows.Count
            If        rMissions(lMission2,        COL_MISSION_CLASS)        =
wsGCS.Range("LINK").Value Then
              Print #pFN, "      m_" & missionNames.Item(lMission) & ".m_" &
rMissions(lMission2, COL_MISSION_NAME)
            End If
          Next lMission2
      nextMissionGCS:
        Next lMission
        If numPreq = 0 Then
          Print  #pFN,  "          m_"  &  missionNames.Item(1)  &  ".m_"  &
missionNames.Item(1)
          Print #hpFN, " m_" & missionNames.Item(1)
        End If
        Print #pFN, "   /"
        Print #pFN, " ;"
```

```
        Close pFN
        Print #hpFN, "   /"
        Print #hpFN, " ;"
        Close hpFN

        Dim mgFN As Integer
        mgFN = FreeFile()
        Open "mg.gms" For Output As mgFN
        Print #mgFN, " SET mg(m,g) /"
        For lMission = 1 To missionNames.Count
          For iRow = 1 To rGCSs.Rows.Count
            lGCS = gcsNames.Lookup(rGCSs(iRow, COL_GCS_NAME))
            If dDistGM(lGCS, lMission) <= rGCSs(iRow, COL_GCS_LOS) Then
              Print #mgFN, "     m_" & missionNames.Item(lMission) & ".g_" &
gcsNames.Item(lGCS)
            End If
          Next iRow
        Next lMission
        Print #mgFN, " /;"
        Close mgFN

        Dim mhFN As Integer
        mhFN = FreeFile()
        Open "mh.gms" For Output As mgFN

        ReDim bClassActive(classNames.Count) As Boolean
        For iRow = 1 To rClasses.Rows.Count
          lClass = classNames.Lookup(rClasses(iRow, COL_MCLASS_NAME))
          If CLng(rClasses(iRow, COL_MCLASS_ACTIVE)) = 1 Then
            bClassActive(lClass) = True
          Else
            bClassActive(lClass) = False
          End If
        Next iRow

        Print #mhFN, " SET mh(m,h) /"
        For iRow = 1 To rMissions.Rows.Count
          lClass = classNames.Lookup(rMissions(iRow, COL_MISSION_CLASS))
          If bClassActive(lClass) Then
```

```
            Print #mhFN, "      m_" & rMissions(iRow, COL_MISSION_NAME) &
".h_active"
          Else
            Print #mhFN, "      m_" & rMissions(iRow, COL_MISSION_NAME) &
".h_passive"
          End If
        Next iRow
        Print #mhFN, " /;"
        Close mhFN

        Dim gvFN As Integer
        gvFN = FreeFile()
        Open "gv.gms" For Output As mgFN
        Print #gvFN, " SET GV(g,v) /"
        For iRow = 1 To rGCSs.Rows.Count
          lGCS = gcsNames.Lookup(rGCSs(iRow, COL_GCS_NAME))
          lTier = tierNames.Lookup(rGCSs(iRow, COL_GCS_TIER))
          Print #gvFN, "        g_" & gcsNames.Item(lGCS) & ".v_" &
tierNames.Item(lTier)
        Next iRow
        Print #gvFN, " /;"
        Close gvFN

        Print #uasDataFN, " r(m) /"
        For iRow = 1 To rMissions.Rows.Count
          lMission = missionNames.Lookup(rMissions(iRow, COL_MISSION_NAME))
          If Trim(CStr(rMissions(iRow, COL_MISSION_REQUIRED))) <> "" Then
            If       CLng(rMissions(iRow,       COL_MISSION_REQUIRED))       >
CLng(rMissions(iRow, COL_MISSION_PERIODS)) Then
              If  MsgBox("Required  coverage  exceeds  total  mission  time",
vbOKCancel) = vbCancel Then
                wsMissions.Activate
                rMissions(iRow, COL_MISSION_REQUIRED).Select
                Exit Sub
              Else
                Print #uasDataFN, "   m_" & missionNames.Item(lMission) & "  "
& CLng(rMissions(iRow, COL_MISSION_PERIODS))
              End If
            Else
              Print #uasDataFN, "   m_" & missionNames.Item(lMission) & "  " &
CLng(rMissions(iRow, COL_MISSION_REQUIRED))
            End If
```

```
        Else
          Print #uasDataFN, "   m_" & missionNames.Item(lMission) & "  0"
        End If
      Next iRow
      Print #uasDataFN, " /"


    '   Count number of UAVs in each tier at each LRS
      ReDim tierAtLRS(tierNames.Count, lrsNames.Count) As Long
      For lLRS = 1 To lrsNames.Count
        For lTier = 1 To tierNames.Count
          tierAtLRS(lTier, lLRS) = 0
        Next lTier
      Next lLRS

      For lUAV = 1 To rUAVs.Rows.Count
        i = tierNames.Lookup(rUAVs(lUAV, COL_UAV_TIER))
        j = lrsNames.Lookup(rUAVs(lUAV, COL_UAV_LRS))
        tierAtLRS(i, j) = tierAtLRS(i, j) + 1
      Next lUAV

    '    a(m,s,t)           indicator if mission m covered by schedule s in
period t
    '   b(v,s,t)         indicator if UAV in Tier v required by schedule s in
period t
    '   f(g,s,t,h)      indicator if gcs g used by schedule s in period t for
mission type h
    '   k(l,s,t)        indicator if lrs l used by schedule s in period t
    '   q(p,s,t)        indicator if payload p used by schedule s in period t

      Dim lMaxRoutes As Long
      lMaxRoutes = CLng(wsDashboard.Range("MAX_ROUTES"))
      Dim lMaxMissions As Long
      lMaxMissions = CLng(wsDashboard.Range("MAX_MISSIONS")) + 1

      ReDim a(lMaxRoutes, wsDashboard.Range("LAST_PERIOD")) As Long
      ReDim b(lMaxRoutes) As Long
      ReDim f_act(lMaxRoutes, wsDashboard.Range("LAST_PERIOD")) As Long
      ReDim f_pas(lMaxRoutes, wsDashboard.Range("LAST_PERIOD")) As Long
      ReDim k(lMaxRoutes, wsDashboard.Range("LAST_PERIOD")) As Long
```

```
        ReDim q(lMaxRoutes, classNames.Count) As Long

        ReDim stack(lMaxMissions + 1) As Long
        ReDim onStack(lNumMissions) As Long
        ReDim dwell(lMaxMissions + 1) As Long

        For lMission = 1 To lNumMissions
          onStack(lMission) = 0
        Next lMission

        Dim top As Long
        top = 0

        lNumSchedules = 0
        lNumLRSSchedules = 0
        bNextRoute = True
        lLRS = 1
        Dim s As Long, t As Long
        Dim lNextMission As Long, lCurrMission As Long
        ReDim lNextStack(lMaxMissions + 1) As Long
        Dim dRemTime As Double
        Dim lRemPeriods As Long
        Dim dTransitTime As Double
        Dim lTransitPeriods As Long
        Dim lcurrNumClasses As Long
        Dim dTierSpeed As Double
        Dim lThisLaunchPeriod As Long
        Dim lThisRecoverPeriod As Long
        Dim lMaxClasses As Long
        ReDim lTransitPeriod(lMaxMissions + 1) As Long
        ReDim lIdlePeriod(lMaxMissions + 1) As Long
        ReDim lMissionPeriod(lMaxMissions + 1) As Long
        ReDim lStartPeriod(lNumMissions) As Long
        ReDim lFinishPeriod(lNumMissions) As Long
        For lMission = 1 To rMissions.Rows.Count
          lStartPeriod(missionNames.Lookup(rMissions(lMission,
COL_MISSION_NAME))) = CLng(rMissions(lMission, COL_MISSION_START_PERIOD))
          lFinishPeriod(missionNames.Lookup(rMissions(lMission,
COL_MISSION_NAME))) = CLng(rMissions(lMission, COL_MISSION_FINISH_PERIOD))
        Next lMission
```

```
        ReDim dMissionValue(missionNames.Count) As Double
        For lMission = 1 To rMissions.Rows.Count
          dMissionValue(missionNames.Lookup(rMissions(lMission,
COL_MISSION_NAME))) = CDbl(rMissions(lMission, COL_MISSION_VALUE))
        Next lMission
        ReDim lHasClass(lNumClasses) As Long
        For lClass = 1 To lNumClasses
          lHasClass(lClass) = 0
        Next lClass
        lcurrNumClasses = 0
        ReDim reqClass(lNumMissions)
        For lMission = 1 To rMissions.Rows.Count
          reqClass(missionNames.Lookup(rMissions(lMission,  COL_MISSION_NAME)))
= _
           classNames.Lookup(rMissions(lMission, COL_MISSION_CLASS))
        Next lMission
        ReDim lClassTierAvail(classNames.Count, tierNames.Count) As Long
        For lClass = 1 To classNames.Count
          For lTier = 1 To tierNames.Count
            lClassTierAvail(lClass, lTier) = 0
          Next lTier
        Next lClass
        For lPayload = 1 To rPayloads.Rows.Count
          lClass = classNames.Lookup(rPayloads(lPayload, COL_PAYLOAD_CLASS))
          lTier = tierNames.Lookup(rPayloads(lPayload, COL_PAYLOAD_TIER))
          lClassTierAvail(lClass,     lTier)     =     CLng(rPayloads(lPayload,
COL_PAYLOAD_QUANTITY))
        Next lPayload
        Dim aFN As Integer, bFN As Integer, fFN As Integer, kFN As Integer, qFN
As Integer
        aFN = FreeFile()
        Open ThisWorkbook.path & "\a.gms" For Output As aFN
        bFN = FreeFile()
        Open ThisWorkbook.path & "\b.gms" For Output As bFN
        kFN = FreeFile()
        Open ThisWorkbook.path & "\k.gms" For Output As kFN
        qFN = FreeFile()
        Open ThisWorkbook.path & "\q.gms" For Output As qFN

        s = 0
        Print #aFN, " SET A(s,m,t) /"
```

```
        Print #bFN, " SET B(s,v,t) /"
        Print #qFN, " SET Q(s,c,t) /"

        Dim tierLRScombos As Long
        tierLRScombos = 0
        For lTier = 1 To tierNames.Count
          For lLRS = 1 To lrsNames.Count
            If tierAtLRS(lTier, lLRS) Then
              tierLRScombos = tierLRScombos + 1
            End If
          Next lLRS
        Next lTier
        Dim currCombo As Long
        currCombo = 1

        For lLRS = 1 To rLRSs.Rows.Count
          Print #logFN, "LRS " & rLRSs(lLRS, COL_LRS_NAME) & ":" &
lrsNames.Lookup(rLRSs(lLRS, COL_LRS_NAME))
          For lTier = 1 To rTiers.Rows.Count
            Print #logFN, " tier " & rTiers(lTier, COL_TIER_NAME) & ":" &
tierNames.Lookup(rTiers(lTier, COL_TIER_NAME))
            If tierAtLRS(tierNames.Lookup(rTiers(lTier, COL_TIER_NAME)),
lrsNames.Lookup(rLRSs(lLRS, COL_LRS_NAME))) > 0 Then
              dTierSpeed = rTiers(lTier, COL_TIER_SPEED)
              dRemTime = rTiers(lTier, COL_TIER_ENDURANCE) '* rTiers(lTier,
COL_TIER_SPEED)
              lRemPeriods = Ceiling(dRemTime * lPeriodsPerHour)
              lMaxClasses = rTiers(lTier, COL_TIER_CAPACITY)
              Print #logFN, " tierAtLRS? yes lMaxClasses=" & lMaxClasses
              For lClass = 1 To lNumClasses
                lHasClass(lClass) = 0
              Next lClass
              lcurrNumClasses = 0

              top = 1
              stack(top) = EVENT_LR 'L/R event at lLRS
              lNextStack(top) = 1
              Do
                lCurrMission = stack(top)
                lNextMission = lNextStack(top)
                'Find next mission to put on the stack
```

```
                Do While lNextMission <= lNumMissions
                  If onStack(lNextMission) > 0 Then
                    'Print #logFN, "  onstack"
                    GoTo skipMission
                  End If
                  If     dDistLM(lLRS,     lNextMission)     >     rTiers(lTier,
COL_TIER_RADIUS) Then
                    'Print #logFN, "Out of range"
                    GoTo skipMission
                  End If
                  If lClassTierAvail(reqClass(lNextMission), lTier) = 0 Then
                    'Print #logFN, "No Payloads available for this Tier"
                    GoTo skipMission
                  End If
                  If lCurrMission = EVENT_LR Then
                    dTransitTime = dDistLM(lLRS, lNextMission) / dTierSpeed
                  Else
                    dTransitTime   =   dDistMM(lCurrMission,   lNextMission)   /
dTierSpeed
                  End If
                  lTransitPeriods = Ceiling(dTransitTime * lPeriodsPerHour)
                  If lCurrMission <= 0 Then
                    If              lFinishPeriod(lNextMission)              <=
lStartPeriod(lCurrMission) + lTransitPeriods Then
                      'Print #logFN, "  time warp"
                      GoTo skipMission
                    End If
                  ElseIf             dMissionValue(lNextMission)             <=
dMissionValue(lCurrMission) Then
                    If              lFinishPeriod(lNextMission)              <=
lFinishPeriod(lCurrMission) + lTransitPeriods Then
                      'Print #logFN, "  time warp"
                      GoTo skipMission
                    End If
                  Else
                    If lStartPeriod(lNextMission) <= lStartPeriod(lCurrMission)
+ lTransitPeriods Then
                      'Print #logFN, "  time-value warp"
                      GoTo skipMission
                    End If
                  End If
```

```
                If lTransitPeriods + Ceiling(lPeriodsPerHour * (dDistLM(lLRS,
lNextMission) / rTiers(lTier, COL_TIER_SPEED))) > lRemPeriods Then
                  'Print #logFN, "  out of time"
                  GoTo skipMission
                End If
                If       (lcurrNumClasses       =       lMaxClasses       And
lHasClass(reqClass(lNextMission)) = 0) Then
                  'Print #logFN, "  no payload space left"
                  GoTo skipMission
                End If
                GoTo foundMission
      skipMission:
                lNextMission = lNextMission + 1
              Loop
      foundMission:
              If lNextMission > lNumMissions Or top >= lMaxMissions Then 'out
of missions: pop stack
                If lCurrMission > 0 Then
                  lHasClass(reqClass(lCurrMission))                         =
lHasClass(reqClass(lCurrMission)) - 1
                  If lHasClass(reqClass(lCurrMission)) = 0 Then
                    lcurrNumClasses = lcurrNumClasses - 1
                  End If
                  onStack(lCurrMission) = 0
                End If
                top = top - 1
                GoTo nextMission
              End If
              If lCurrMission = EVENT_LR Then
                lTransitPeriod(top) = lTransitPeriods
                lThisLaunchPeriod     =     lStartPeriod(lNextMission)     -
lTransitPeriods - 1
                If lThisLaunchPeriod < 1 Then
                  lThisLaunchPeriod = 1
                End If
                lRemPeriods   =   Ceiling(dRemTime   *   lPeriodsPerHour)   +
lThisLaunchPeriod - 1
                lTransitPeriod(top) = lThisLaunchPeriod + lTransitPeriods
                lIdlePeriod(top) = lThisLaunchPeriod - 1
              Else
                lMissionPeriod(top) = lFinishPeriod(lCurrMission)
```

```
                If dMissionValue(lCurrMission) < dMissionValue(lNextMission)
And _
                   lStartPeriod(lNextMission) - lTransitPeriods - 1 <
lFinishPeriod(lCurrMission) Then
                  lMissionPeriod(top) = lStartPeriod(lNextMission) -
lTransitPeriods - 1
                End If
                lTransitPeriod(top) = lMissionPeriod(top) + lTransitPeriods
                lIdlePeriod(top) = lStartPeriod(lNextMission) - 1
              End If
              lNextStack(top) = lNextMission + 1
              top = top + 1
              stack(top) = lNextMission
              onStack(lNextMission) = top
              lMissionPeriod(top) = lFinishPeriod(lNextMission)
              lTransitPeriod(top) = lFinishPeriod(top) +
Ceiling((dDistLM(lLRS, lNextMission) / rTiers(lTier, COL_TIER_SPEED)) *
lPeriodsPerHour)
              lNextStack(top) = 1
              If lHasClass(reqClass(lNextMission)) = 0 Then
                lcurrNumClasses = lcurrNumClasses + 1
              End If
              lHasClass(reqClass(lNextMission)) =
lHasClass(reqClass(lNextMission)) + 1


              'New feasible schedule: now copy stack info into appropriate
arrays
              s = s + 1
              t = 1
              'calculate start period of mission sequence: find start time of
first mission
              ' in stack(2), figure out latest departure period to arrive at
first mission (if
              ' earlier than period 1, then l/r in period 1, transit, and do
as much of mission
              ' as possible
              Print #logFN, "dTransitTime " & dTransitTime & "
lTransitPeriods " & lTransitPeriods
              Print #logFN, "s:" & CStr(s) & " ";
              For iRow = 1 To top
                Print #logFN, stack(iRow) & ",";
              Next iRow
              Print #logFN, "0"
```

```
Print #logFN, "m:" & CStr(s) & " ";
For iRow = 1 To top
  Print #logFN, lMissionPeriod(iRow) & ",";
Next iRow
Print #logFN, "0"
Print #logFN, "t:" & CStr(s) & " ";
For iRow = 1 To top
  Print #logFN, lTransitPeriod(iRow) & ",";
Next iRow
Print #logFN, "0"
Print #logFN, "i:" & CStr(s) & " ";
For iRow = 1 To top
  Print #logFN, lIdlePeriod(iRow) & ",";
Next iRow
Print #logFN, "0"

Do While t < lThisLaunchPeriod And t <= lLastPeriod
  a(s, t) = EVENT_IDLE
  t = t + 1
Loop
a(s, t) = EVENT_LR
f_act(s, t) = 1
k(s, t) = lLRS
t = t + 1
Do While t <= lTransitPeriod(1) And t <= lLastPeriod
  a(s, t) = EVENT_TRANSIT
  f_pas(s, t) = 1
  t = t + 1
Loop
For iRow = 2 To top
  lCurrMission = stack(iRow)
  If iRow < top Then
    lNextMission = stack(iRow + 1)
  Else
    lNextMission = EVENT_LR 'L/R event
  End If
  ' XXX Need upper bounds: time horizon, flight time, etc...
  Do While t <= lMissionPeriod(iRow) And t <= lLastPeriod
    'fill in arrays
```

```
                    a(s, t) = lCurrMission
                    f_act(s, t) = 1
                    t = t + 1
                  Loop
                  Do While t <= lTransitPeriod(iRow) And t <= lLastPeriod
                    a(s, t) = EVENT_TRANSIT
                    f_pas(s, t) = 1
                    t = t + 1
                  Loop
                  Do While t <= lIdlePeriod(iRow) And t <= lLastPeriod
                    a(s, t) = EVENT_IDLE
                    f_pas(s, t) = 1
                    t = t + 1
                  Loop
                Next iRow

                If t <= lLastPeriod Then
                  a(s, t) = EVENT_LR
                  lThisRecoverPeriod = t
                  f_act(s, t) = 1
                  k(s, t) = lLRS
                  b(s) = lTier
                End If
                For iRow = 1 To lLastPeriod
                  Print #logFN, ":" & a(s, iRow);
                Next iRow
                Print #logFN, ""
                For iRow = 2 To top
                  For lPeriod = 1 To lLastPeriod
                    If a(s, lPeriod) = stack(iRow) Then
                      Print #aFN, "            s_" & CStr(s) & ".m_" &
missionNames.Item(stack(iRow)) & ".t_" & CStr(lPeriod)
                    End If
                  Next lPeriod
                Next iRow
                For lPeriod = 1 To lLastPeriod
                  If lPeriod >= lThisLaunchPeriod And lPeriod <=
lThisRecoverPeriod Then
                    Print #bFN, "            s_" & CStr(s) & ".v_" &
Trim(rTiers(lTier, COL_TIER_NAME)) & ".t_" & CStr(lPeriod)
```

```
                End If
              Next lPeriod
              Print #kFN, " k('s_" & CStr(s) & "','l_" & Trim(rLRSs(lLRS,
COL_LRS_NAME)) & "','t_" & CStr(lThisLaunchPeriod) & "')=1;"
              Print #kFN, " k('s_" & CStr(s) & "','l_" & Trim(rLRSs(lLRS,
COL_LRS_NAME)) & "','t_" & CStr(lThisRecoverPeriod) & "')=1;"
              For lClass = 1 To classNames.Count
                If lHasClass(lClass) > 0 Then
                  q(s, lClass) = 1
                  For lPeriod = 1 To lLastPeriod
                    If   lPeriod   >=   lThisLaunchPeriod   And   lPeriod   <=
lThisRecoverPeriod Then
                      Print   #qFN,   "           s_"   &   CStr(s)   &   ".c_"   &
classNames.Item(lClass) & ".t_" & CStr(lPeriod)
                    End If
                  Next lPeriod
                End If
              Next lClass
              If s >= lMaxRoutes Then GoTo doneMissions
              If s Mod 1000 = 0 Then wsDashboard.Range("ROUTESCELL") = s
              If  s  >=  (currCombo  -  1)  *  lMaxRoutes  /  (tierLRScombos)  Then
GoTo nextTier
       nextMission:
              Loop While top > 0
            End If
       nextTier:
            currCombo = currCombo + 1
          Next lTier
        Next lLRS

       doneMissions:
        wsDashboard.Range("ROUTESCELL") = s
        Print #uasSetsFN, " s /s_1*s_" & CStr(s) & " /"
        Print #uasSetsFN, " ;"
        Print #uasDataFN, " ;"
        Print #aFN, "   /"
        Print #aFN, " ;"
        Print #bFN, "   /"
        Print #bFN, " ;"
        Print #qFN, "   /"
        Print #qFN, " ;"
```

```
subExit:
  Print #logFN, "Ending output"
  Close uasSetsFN
  Close uasDataFN
  Close aFN
  Close bFN
  Close kFN
  Close qFN
  Close logFN
End Sub
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C GAMS CODE

```
$TITLE UAS Planner 08.06.02

$ontext
MCCDC/OAD UAS Mission Planner
Assigns a fleet of UAVs to a list of missions with various
requirements and time windows over a fixed time horizon of many
time periods.
Solved using explicit column generation, one column per UAS schedule.
$offtext
$inlinecom { }
 FILE UASP_log/UASP.log/ ;
 PUT UASP_log ;
 PUT 'UASP - MCCDC/OAD UAS Planner 08.06.02' // ;
 PUTCLOSE UASP_log ;

 FILE UASP_sta/UASP.sta/ ;
 PUT UASP_sta ;
 PUT 'ERROR - Did not solve' // ;
 PUTCLOSE UASP_sta ;

 FILE VALUE_CSV/VALUE.csv/ ;
 PUT VALUE_CSV ;
 PUT ' ' / ;         { scratch legacy file contents }
 PUTCLOSE VALUE_CSV ;

 FILE MISSIONS_CSV/MISSIONS.csv/ ;
 PUT MISSIONS_CSV ;
 PUT ' ' / ;         { scratch legacy file contents }
 PUTCLOSE MISSIONS_CSV ;
```

```
FILE SCHEDULES_CSV/SCHEDULES.csv/ ;
PUT SCHEDULES_CSV ;
PUT ' ' / ;         { scratch legacy file contents }
PUTCLOSE SCHEDULES_CSV

FILE DUALS_CSV/DUALS.csv/ ;
PUT DUALS_CSV ;
PUT ' ' / ;         { scratch legacy file contents }
PUTCLOSE DUALS_CSV

PUT UASP_log ;
UASP_log.ap=1 ;     { re-open log file, appending to prior contents }

OPTIONS
  optcr    = 0.1
  limrow   =   0
  limcol   =   0
  SEED     = 1234
  ITERLIM  = 10000000
  LP       = CPLEX
  RMIP     = CPLEX
  MIP      = CPLEX
;
$ONTEXT
{STATIC, GROUNDED} SETS
  v          Tier levels
  h          mission types          {active, passive}
  g          Ground control stations
  l          Launch and recovery sites
  m          Missions               {missions that ships can be assigned}
  c          Payload classes
```

```
  t           Time periods
  s           UAV emplyment schedules
;
$OFFTEXT
$INCLUDE uasSets.gms

alias(m,mp);
$ONTEXT
{GIVEN} PARAMETERS
  num_uavs(l,v,t)   number of uavs available in tier v in period t
  num_mods(c,t)     number of payload modules available of class c in period t
  gcs_cap(g,t,h)    max num UAVs on gcs g in period t on mission type h
  lrs_cap(l,t)      max num UAVs at LRS l in period t
  length(m)         maximum periods of coverage of mission m
  val(m)            value of mission m
  r(m)              required periods of coverage for mission m
  k(s,l,t)          indicator if lrs l used by schedule s in period t
;

$OFFTEXT
$OFFLISTING
$INCLUDE uasData.gms
PARAMETER
  k(s,l,t)
;
$INCLUDE k.gms

$ONTEXT
{DYNAMIC} SETS
  A(s,m,t)        schedule s covers mission m in period t
  B(s,v,t)        schedule s uses tier v UAV in period t
  Q(s,c,t)        schedule s requires payload class c in period t
```

```
   P(m,mp)        mission prerequisite fixed mission m requires at least one
                  mission mp in same period
   E(m,mp)        exclusive missions m and mp cannot be accomplished in same
                  period
  ;
  {Notional data: will be replaced with INCLUDE statements and accompanying
data files}
  $OFFTEXT
  alias(m,mp);
  alias(v,vp);
  alias(t,tp);
  $INCLUDE a.gms
  $INCLUDE b.gms
  $INCLUDE q.gms
  $INCLUDE hp.gms
  $INCLUDE p.gms
  $INCLUDE mg.gms
  $INCLUDE mh.gms
  $INCLUDE gv.gms
  $ONLISTING

  SET sl(s,l);
  LOOP((s,l)$(SUM(t,k(s,l,t))>eps),
   sl(s,l) = yes;
  );

  SCALARS errors,warnings ;

  errors=0 ;
  warnings=0;
  {Loops for data checking go here}
  IF(errors>0,
    PUT / 'inconsistencies found in active and value data: ',errors:10:0 / ;
```

```
  PUT '(should I have shut you down here?' / ;
);

VARIABLE
  OBJECTIVE
;
BINARY VARIABLES
  X(s)          Schedule s flown by some UAV
  Y(m,v,t)      Mission m covered in period t
  W(m,v,g,t)    Mission m handled by GCS g in period t
;
INTEGER VARIABLES
  D(m)          Total number of dwell periods on mission m
  MODS(c,v,l)
;
EQUATIONS
  eqnObj              {R0} Objective function measures total value of mission
                      accomplishment
  UAVLimit(l,v,t)     {R1} limit on UAVs flying by tier and period
  ModuleLimit(c,v,l,t) {R2} limit on payload modules flying by payload and
period
  TotalMods(c,v)      {R2.5}
  GCSLimit(g,t,h)     {R3} limit on GCS use by gcs period and type
  LRSLimit(l,t)       {R4} limit on LRS use by LRS and period
  CoverageLimit(m,v,t)  {R5} mission and period coverage control
  DwellLimit(m)       {R6} limit on total dwell time by mission
  Prerequisite(m,v,t)   {R7} prerequisite controls by period
  Mission_Control(m,v,t)
* Exclusive(m,mp,t)    {R8} mutually exclusive missions by period
  RequiredDwell(m)    {R9} required missions must be completely covered
;
```

```
eqnObj..                    {R0}
  OBJECTIVE =e= sum(m, val(m)*D(m))
;
UAVLimit(l,v,t)..           {R1}
  SUM(s$(B(s,v,t) and sl(s,l)), X(s)) =l= num_uavs(l,v,t)
;
ModuleLimit(c,v,l,t)..      {R2}
  SUM(s$(sl(s,l) and B(s,v,t) and Q(s,c,t)), X(s)) =l= MODS(c,v,l)
;
TotalMods(c,v)..
  SUM(l, MODS(c,v,l)) =l= num_mods(c,v)
;
GCSLimit(g,t,h)..           {R3}
  SUM((m,v)$(MG(m,g) and MH(m,h) and GV(g,v)), W(m,v,g,t)) =l= gcs_cap(g,t,h)
;
LRSLimit(l,t)..             {R4}
  SUM(s$(k(s,l,t)>eps), X(s)) =l= lrs_cap(l,t)
;
CoverageLimit(m,v,t)..      {R5}
  Y(m,v,t) =l= SUM(s$(A(s,m,t) and B(s,v,t)), X(s))
;
DwellLimit(m)..             {R6}
  D(m) =l= SUM((v,t), Y(m,v,t))
;
Prerequisite(m,v,t)$HP(m).. {R7}
  Y(m,v,t) =l= SUM((mp,vp)$P(m,mp),Y(mp,vp,t))
;
Mission_Control(m,v,t)$(not HP(m))..
  Y(m,v,t) =l= SUM(g$(MG(m,g) and GV(g,v)), W(m,v,g,t))
;
* Exclusive(m,mp,t)$E(m,mp)..  {R8}
```

```
*   Y(m,t) + Y(mp,t) =l= 1
* ;
 RequiredDwell(m)$(r(m)>eps)..          {R9}
  D(m) =g= r(m)
 ;
 LOOP(m,
  D.up(m) = length(m);            {R10}
 );

 MODEL UAS_PLANNER /
  eqnObj
  UAVLimit
  ModuleLimit
  TotalMods
  GCSLimit
  LRSLimit
  CoverageLimit
  DwellLimit
  Prerequisite
*  Exclusive
  Mission_Control
  RequiredDwell
 /
 ;

 IF(errors=0,
  PUT 'solve UAS_PLANNER' / ;
*   UAS_PLANNER.reslim=60. ;
*   UAS_PLANNER.optcr=0.05 ;
  PUT ' reslim=',UAS_PLANNER.reslim:5:0,' seconds, optcr=',UAS_PLANNER.optcr:5:2 ;
  PUT '  optfile=',UAS_PLANNER.optfile:2:0 / ;
```

```
SOLVE UAS_PLANNER USING MIP MAXIMIZING OBJECTIVE;
PUT 'PLANNER solver and model status ',UAS_PLANNER.solvestat:4:0,UAS_PLANNER.modelstat:4:0 / ;
IF( UAS_PLANNER.modelstat<>1 and UAS_PLANNER.modelstat<>8,
 errors=1 ;
ELSE
 PUT / 'Objective: ',OBJECTIVE.l:10:4 /;
 PUT / 'Module Deployment' / ;
 LOOP((c,v,l)$(MODS.L(c,v,l)>eps),
  PUT '  ',c.tl:10,' ',v.tl:8,' ',l.tl:10,': ',MODS.L(c,v,l):4:0 / ;
 );
 PUT / 'UAV flight schedules..' // ;
 LOOP(s$(X.l(s)>eps),
  PUT '  ',s.tl:10,'  ' ;
  LOOP(v$SUM(t$B(s,v,t),1),
   PUT '  ',v.tl:8 ;
  );
  PUT '  ',X.l(s):4:1 / ;
  LOOP(m$(SUM(t$A(s,m,t),1)>0),
   PUT '    ',m.tl:10 ;
   LOOP(t$A(s,m,t),
    PUT ' ',t.tl:5 ;
   );
   PUT / ;
  );
 );
 PUT / 'GCS Usage' / ;
 LOOP(g$(SUM((m,v,t)$(MG(m,g) and GV(g,v)),W.l(m,v,g,t))>eps),
  PUT '  ',g.tl:8 / ;
  LOOP((m,v)$(MG(m,g) and GV(g,v) and (SUM(t,W.l(m,v,g,t))>eps)),
   PUT '     ',m.tl:8,'   ',(SUM(t,W.l(m,v,g,t))) ;
   LOOP(t$(W.l(m,v,g,t)>eps),
```

```
          PUT ' ',t.tl:8 ;
        );
        PUT / ;
      );
    );
    PUT / 'MISSION COVERAGE' / ;
    PUT '   Mission    Dwell Time' / ;
    LOOP(m$(D.l(m)>0),
      PUT '  ',m.tl:8,'  ',D.l(m):5:1 / ;
      LOOP(t$(SUM(v,Y.l(m,v,t))>eps),
        PUT '    ',t.tl:4,' ',(SUM(v,Y.l(m,v,t))):4:1 / ;
      );
    );
    PUTCLOSE UASP_LOG;

    PUT MISSIONS_CSV ;
    PUT ',,,Total,Total,Value,Total' /;
    PUT ',Start,End,Time Periods,Time Periods,Per Time,Value' /;
    PUT
'Mission,Time,Time,Requested,Assigned,Period,Achieved,',OBJECTIVE.L /;
    LOOP(m,
      PUT m.tl,',,,,',D.l(m),',',val(m),',',(D.l(m)*val(m)) /;
    );
    PUTCLOSE MISSIONS_CSV;
     {Rest of report writer goes here}
  );
);

IF(errors>0,
  PUT ' <*** errors aborting this run:',errors:10:0 / ;
);
IF(warnings>0,
```

```
  PUT / ' <+++ warnings generated by this run: ',warnings:10:0 / ;
  PUT '  please find out what these indicate' / ;
);
PUTCLOSE UASP_log ;

IF(errors=0,

  {Build CSV Output Files}

  PUT UASP_STA;
  PUT 'Optimal solution found. Successful completion.' /;
  PUTCLOSE UASP_STA;
);
```

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Operation Analysis Division C19
   Quantico, VA

4. Marine Corps Representative
   Naval Post Graduate School
   Monterey, CA

5. Director, Training and Education, MCCDC, Code C46
   Quantico, VA

6. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, VA

7. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer)
   Camp Pendleton, California

8. Director, Studies and Analysis Division, MCCDC, Code C45
   Quantico, VA