
# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

# THESIS

**OPTIMIZING THE U.S. MARINE CORPS' SELECTIVE REENLISTMENT BONUS PROGRAM FOR CAREER FORCE RETENTION**

by

Kent A. Robbins, Jr.

September 2007

| | |
|---|---|
| Thesis Advisor: | R. Kevin Wood |
| Second Reader: | Ronald D. Fricker, Jr. |

**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** September 2007 | **3. REPORT TYPE AND DATES COVERED** Master's Thesis | |
| **4. TITLE AND SUBTITLE** Optimizing the U.S. Marine Corps' Selective Reenlistment Bonus Program for Career Force Retention | | | **5. FUNDING NUMBERS** |
| **6. AUTHOR(S)** Kent A. Robbins, Jr. | | | |
| **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** A |


**13. ABSTRACT (maximum 200 words)**

The Marine Corps uses its Selective Reenlisted Bonus (SRB) Program to influence Marines to reenlist for a designated term into certain Military Occupational Specialties (MOSs) in order to reach planned manpower goals. The bonus amount is determined by selecting an "SRB multiplier" for each combination of MOS and Zone ("MOSZ"). ("Zone" corresponds to length of service.) A higher multiplier means a larger bonus and leads to a higher percentage of Marines reenlisting. That percentage, predicted by an existing forecasting model, is assumed exact here.

The "SRB multiplier model" assigns multipliers to minimize a sum of weighted squared deviations from MOSZ targets subject to a budget constraint. This model is implemented as a generalized assignment problem, and solved approximately on a personal computer using Lagrangian relaxation and a secondary heuristic. (The algorithm is programmed in Visual Basic for Applications and has an Excel interface.)

Data for FY04 shows 491 bonus-eligible MOSZs. With up to 11 possible multiplier values, this yields a model with 5,401 0-1 variables and 491 constraints. A solution within 0.0018% of optimality is reached in 1.4 seconds on 1.58 GHz personal computer. Standard integer-programming software verifies the correctness of the solution.


| **14. SUBJECT TERMS** Selective Reenlistment Bonus, Multiplier, Generalized Assignment Problem, Lagrangian Relaxation | | | **15. NUMBER OF PAGES** 87 |
|---|---|---|---|
| | | | **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 THE U.S. MARINE CORPS' SELECTIVE REENLISTMENT BONUS PROGRAM FOR THE CAREER FORCE RETENTION**

Kent A. Robbins, Jr.
Captain, United States Marine Corps
B.S., United States Naval Academy, 2001

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN OPERATIONS RESEARCH**

from the

**NAVAL POSTGRADUATE SCHOOL**
**September 2007**

Author: Kent A. Robbins, Jr.

Approved by: R. Kevin Wood
Thesis Advisor

Ronald D. Fricker, Jr.
Second Reader

James N. Eagle
Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

The Marine Corps uses its Selective Reenlisted Bonus (SRB) Program to influence Marines to reenlist for a designated term into certain Military Occupational Specialties (MOSs) in order to reach planned manpower goals. The bonus amount is determined by selecting an "SRB multiplier" for each combination of MOS and Zone ("MOSZ"). ("Zone" corresponds to length of service.) A higher multiplier means a larger bonus and leads to a higher percentage of Marines reenlisting. That percentage, predicted by an existing forecasting model, is assumed exact here.

The "SRB multiplier model" assigns multipliers to minimize a sum of weighted squared deviations from MOSZ targets subject to a budget constraint. This model is implemented as a generalized assignment problem, and solved approximately on a personal computer using Lagrangian relaxation and a secondary heuristic. (The algorithm is programmed in Visual Basic for Applications and has an Excel interface.)

Data for FY04 shows 491 bonus-eligible MOSZs. With up to 11 possible multiplier values, this yields a model with 5,401 0-1 variables and 491 constraints. A solution within 0.0018% of optimality is reached in 1.4 seconds on 1.58 GHz personal computer. Standard integer-programming software verifies the correctness of the solution.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| CNA | Center for Naval Analysis |
| CFRM | Career Force Retention Model |
| DOD | Department of Defense |
| EAS | End of Active Service |
| FTAP | First Term Alignment Plan |
| FY | Fiscal Year |
| GAMS | General Algebraic Modeling System |
| GAR | Grade Adjusted Recapitulation |
| LatMove | Lateral Move |
| M&RA | Manpower and Reserve Affairs, Headquarters Marine Corps |
| MOS | Military Occupational Specialty |
| MOSZ | Military Occupational Specialty and Service Time Zone Combination |
| OccField | Occupational Field |
| ROR | Rate of Return |
| SRB | Selective Reenlistment Bonus |
| STAP | Subsequent Term Alignment Plan |
| USMC | United States Marine Corps |
| VBA | Visual Basic for Applications |
| Zone | Service Time Zone |

THIS PAGE INTENTIONALLY LEFT BLANK

# EXECUTIVE SUMMARY

The United States Marine Corps (USMC) uses its Selective Reenlisted Bonus (SRB) Program to influence Marines to reenlist for a designated term into certain Military Occupational Specialties (MOS) in order to reach planned manpower goals. The amount of the bonus is determined by selecting an SRB multiplier for each combination of MOS and Zone ("MOSZ"). ("Zone" partitions Marines into three groups based on current length of service.) A higher multiplier means a larger bonus and leads to a higher percentage of eligible Marines reenlisting. That percentage is predicted by an existing forecasting model, whose predictions are assumed exact.

The total amount of SRB funding available for a fiscal year (FY) is limited by the amount approved and allocated in the FY Defense Budget. Thus, not all MOSZs can be assigned the multiplier necessary to reach reenlistment goals. Hence, it is important to allocate the available SRB funding to those MOSZs that are critical to accomplishing the Marine Corps' mission. So, the problem that is addressed by this thesis is to determine the optimal set of multipliers to reach reenlistment goals.

The "SRB multiplier model" assigns multipliers to minimize a sum of weighted squared deviations from MOSZ targets subject to a budget constraint. This was described and implemented as a generalized assignment problem in a 1986 Naval Postgraduate School thesis by U.S. Marine Corps Captain Dean D. DeWolfe, entitled *Determination of Selective Reenlistment Bonus Multipliers in the United States Marine Corps*. This model has not been used by the USMC, however, because it is no longer current with respect to SRB rules and parameter definitions, and because it was not implemented in a user-friendly programming environment.

This thesis overcomes the earlier problems with the SRB multiplier model by (a) updating parameter definitions and other model constructs to represent the current SRB environment, and (b) creating a flexible implementation of DeWolfe's Lagrangian-based solution algorithm on a personal computer. The flexible implementation should allow the model and solution algorithm to track changes in data and structure for the SRB program as it evolves in the future. The algorithm is programmed in Visual Basic for Applications

and is supplied with a user interface through Microsoft Excel. The algorithm first optimizes a Lagrangian function that relaxes the budget constraint, and saves the best feasible solution it finds during this process. The algorithm then attempts to improve that solution with a greedy marginal-rate-of-return heuristic.

Data for FY04 yields 491 MOSZs that are eligible for an SRB multiplier. With up to 11 multiplier values for some MOSZs, this results in a model with 5,401 0-1 variables and 491 constraints. A solution is reached in 1.4 seconds on a 1.58 GHz personal computer. The solution is within 0.0018% of optimality and consumes 99.99% of the budget.

# ACKNOWLEDGMENTS

First, I would like to offer my heartfelt appreciation to my wife, Ruth, for being able to pick up my slack and keep our growing household running. I would not have been able to complete this without your love and patience.

To Major Dean Conatser, Captain Dave Raymond, and Captain Shaun Doheney: Thank you for your help instructing me on the manpower process and for the quick answers when I needed them.

Last, I would also like to offer my thanks to Professor Kevin Wood and Professor Ron Fricker for their help and encouragement in during the writing of this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

Each year the United States Marine Corps (USMC) must set the Selective Reenlistment Bonus (SRB) for each Military Occupational Specialty (MOS) and Service Time Zone (Zone) (USMC 1990). The goal is to selectively encourage Marines to reenlist and enable the USMC to reach reenlistment targets for the given fiscal year. The purpose of this thesis is to give manpower planners at Manpower & Reserve Affairs (M&RA), Headquarters Marine Corps, a tool that will identify the optimal "SRB multiplier," for each MOS and Zone combination (MOSZ). Theses multipliers set the monetary values of the bonuses.

## A. ENLISTED MARINES, MOS AND ZONE

The Marine Corps uses two models, the First Term Alignment Plan (FTAP) and the Subsequent Term Alignment Plan (STAP), to determine the number of reenlistments that are necessary to meet the future force structure as depicted by the Grade Adjusted Recapitulation (GAR) (Manpower Plans and Policy Branch-Integration and Analysis 2007). The GAR is a planning tool that specifies target enlistment numbers for all MOSs and ranks. For planning purposes, a Marine is classified by pay grade and time in service. Pay grade is associated with a Marine's level of responsibility while time in service correlates to experience. A Private (E-1) is the lowest enlisted pay grade, is the least experienced Marine with less than one year in service, and is generally still in initial training. A Sergeant Major (E-9) is the highest enlisted pay grade and is the most experienced enlisted Marine, having 15 or more years in service.

For reenlistments purposes, manpower planners are concerned with Marines who are in or about to enter the career force, meaning they have 21 or more months time in service. These Marines are placed into one of three Service Time Zones according to time in service as listed in Table 1. The FTAP model covers Marines in Zone A while the STAP model covers Marines in Zones B and C.

Table 1. Service Time Zones designated by time in service.

| ZONE | FROM | TO |
|---|---|---|
| A | 21 Months | 6 Years |
| B | 6 Years | 10 Years |
| C | 10 Years | 14 Years |

FTAP is the retention model used to help determine the number of reenlistments needed from Zone-A Marines, according to MOS, to meet the career-force requirements set by the GAR (USMC 2004). A Zone-A Marine is in his initial enlistment contract, including extensions. The purpose of the FTAP is to fill vacancies in the career force that result from separation or loss for any reason while preventing promotion stagnation and ensuring opportunities for advancement.

STAP is similar to the FTAP, but focuses on subsequent reenlistments of Marines who constitute the career force (Manpower Plans and Policy Branch-Integration and Analysis, 2007). The primary purpose of the STAP is to "move" career-force inventory levels towards required levels. STAP recognizes that any career-force imbalances cannot be "fixed" in one year, and tries to make the following year better in relation to the imbalance in the previous year.

FTAP is important because it covers the most flexible opportunity the Marine Corps has to shape the career force, with respect to the number of Marines in each MOS and pay grade. Only during a Marine's first reenlistment opportunity can the Marine Corps deny reenlistment (without monetary cost) because his particular MOS is full: A reenlistment is allowed only when there is a specific vacancy in the career force that needs to be filled.

Provided that a first-term Marine is qualified, any Marine who is denied reenlistment in his original MOS, because no vacancies exist, is allowed to reenlist in another MOS that needs additional manpower (USMC 1993). This process is called a Lateral Move or "Lat Move." Reenlistments for subsequent-term (Zone-B and Zone-C) Marines are not based on vacancies but instead are based on targets. Furthermore, the

MOS will not close to reenlistments even if the target is reached. Although allowed in specific cases, Lat Moves are not used with subsequent-term Marines as they are with first-term Marines.

The ability to deny first-term reenlistment in conjunction with the use of Lat Moves allows manpower planners to control the number of Marines that advance into the career force from the first-term population, in specific MOSs, and to maintain the personnel goals set forth by the GAR. During subsequent reenlistment opportunities with a career-force Marine, the Marine cannot be denied reenlistment provided the Marine has no performance or criminal-conduct problems and has been promoted on schedule. If, in order to control the MOS population, manpower planners need to separate a subsequent-term Marine who has no difficulties, the affected Marine is entitled to a substantial involuntary-separation allowance.

Historically, most MOSZs do not need a bonus to encourage enough Marines to reenlist to meet assigned targets. Approximately half of the bonus eligible MOSZs will not receive a bonus because the anticipated manpower shortfall is not severe enough to warrant a bonus. For the remaining MOSZs, the manpower shortfall is critical and the only option that can help meet the remaining reenlistment requirements is the SRB.

## B. HOW THE SRB IS USED

When too few Marines want to reenlist or Lat Move into a critical MOSZ, the Marine Corps uses the SRB to increase the number of reenlistments there. Typically, a critical MOS falls into one of the following categories (Morgan 2006):

- Technical skills requiring large training and/or replacement costs,
- Skills in high demand in the civilian sector,
- Skills for which recruiting is difficult,
- Skills that are crucial to combat readiness, and
- Skills in high demand but with a small reenlistment population.

The Marine Corps separates Marines into one of three Service Time Zones by their time in service, as defined in the previous section. An SRB can be awarded to a Marine only once in each Zone, but a Marine may be eligible to receive an SRB in each Zone through his career.

The monetary amount awarded for an SRB is computed as a function of the Marine's pay, additional years of service that reenlistment obligates, and the "strength" of the bonus referred to as an *SRB multiplier*. More precisely, the SRB is defined as

The Minimum of:

Monthly Base Pay × Years of Additional Obligated Service
× SRB Multiplier for MOSZ

and

Maximum Allowable Bonus for the Fiscal Year.

The Marine Corps uses SRB multipliers ranging from 0 to 5 in increments of 0.5. In an MOS that does not need an incentive; the multiplier is 0 and no bonus is offered. In an MOS with an expected shortfall, the multiplier will typically be non-zero.

For Zone A, manpower planners have the power to control the amount of total bonus money that is paid by closing an MOS to reenlistment. Manpower planners cannot, however, close off bonuses or reenlistments to a Zone-B or C MOS, once a bonus is offered there. This means that if all of the eligible Marines in an MOS reenlist, all will receive the bonus money. These reasons combined with the limited yearly budget for the SRB Program make picking the appropriate multiplier values crucial.

## C. CURRENT SRB MODELS

Currently, the SRB Program uses two manual heuristics to specify SRB multiplier values, one heuristic for FTAP (Zone A) and the other for STAP (Zones B and C). The total budget is first divided between FTAP and STAP, and then the heuristics are run separately. Both heuristics are implemented using Excel spreadsheets that first set the multiplier in each MOSZ at the minimum level that is estimated to lead to the meeting of its reenlistment requirement. In order to determine which multiplier to assign, both

models use expected reenlistment rates provided by the Center for Naval Analysis (CNA) (North 1995). Once the initial settings are complete, the expected costs for the selected multipliers are summed over all MOSZs to estimate a total cost of the program. If this estimate exceeds the SRB budget, individual SRB multipliers for less critical MOSs are manually reduced until it does not.

Manpower planners combine the output of FTAP and STAP into a list of prospective SRB multipliers, and carry out an iterative, negotiating process with other departments at Headquarters Marine Corps, to agree upon the final SRB multiplier values. The other departments include the occupational-field sponsors, assignment monitors, command-level representatives, and career-retention specialists, all of whom are the end-users of the SRB Program.

## D. SUMMARY OF PAST WORK ON SRB OPTIMIZATION AND OTHER RELATED RESEARCH

A 1986 Naval Postgraduate School thesis by U.S. Marine Corps Captain Dean D. DeWolfe, entitled *Determination of Selective Reenlistment Bonus Multipliers in the United States Marine Corps*, is the basis for this thesis. First, DeWolfe sets up and defines the problem of selecting an SRB multiplier for each MOSZ as a nonlinear knapsack problem. The objective is to select multipliers that minimize a total weighted deviation from the number of estimated reenlistments versus the reenlistment targets, subject to a budget constraint; all coefficients are expressed as expected values. Next, DeWolfe reformulates the problem as a special generalized assignment problem shows how to maximize a Lagrangian lower bound on the optimal objective value and, in the process, identify a good feasible solution. Finally, due to the nature of the Lagrangian relaxation, it is possible the solution does not use the entire budget. In that case a secondary heuristic tries to increase multipliers to consume it all. The heuristic uses a marginal rate of return to determine which MOS multiplier value can be raised to the next level to achieve the largest decrease in the objective value per unit of (additional) budget consumed.

Although M&RA liked how the model solved the SRB multiplier problem, there were problems that made it unusable. First, the model was written in FORTRAN 77 which is not familiar to personnel at M&RA. Second, there have been many changes to the SRB program rules since 1986 that needed to be updated (SRB multiplier increments, budgets, lump sums, maximum allowed bonuses), and many of these rules were "hard-wired" into the code. Finally, and most importantly, M&RA wanted a model that could run in a familiar computational environment like Excel's VBA (Visual Basic for Applications). Such an environment would allow them to internally use and maintain the model.

Other related research includes two NPS theses by U.S. Marine Corps Captain Jonathan D. Raymond (2006) entitled *Determining the Number of Reenlistments Necessary to Satisfy Future Force Requirements*, and by U.S. Marine Corps Major Dean G. Conatser (2006) entitled *Forecasting U.S. Marine Corps Reenlistments by Military Occupational Specialty and Grade*. Raymond created a model to determine the number of reenlistments that are necessary for each MOS and grade by applying transition rates to the current inventory of Marines who are not eligible for reenlistment and subtracting that amount from the desired force structure in the GAR. Conatser created a model to predict the number of reenlistments for both FTAP and STAP Marines by MOS and grade. These two models, developed at the request of M&RA, are the beginning of the development of the Career Force Retention Model (CFRM) which is being explored to replace the FTAP and STAP models. The CFRM, in conjunction with this thesis, will help manpower planners estimate the required number of new recruits, reenlistments, and the most effective way to set SRB multipliers to influence the reenlistments.

This thesis extends and updates DeWolfe's model and solution algorithm for use by M&RA. This is accomplished by revising budget constraints, adding a Lat Move functionality, and making all model parameters accessible to the user (i.e., no hard-wiring of parameter values into the code). The ultimate goal of this thesis is to create a model that is user-friendly and "user-updatable."

In a paper by DeWolfe, Stevens, and Wood in 1993 entitled *Setting Military Reenlistment Bonus,* the authors expand upon the model that was developed in

DeWolfe's 1986 thesis for use by the United States Army. First a more complicated formulation was created by adding an additional constraint that limited the number of "high-level" bonuses to 10% of the total bonuses offered. Second, the formulation found multipliers for each MOS and Zone based on expected reenlistment rates for 3, 4, 5, and 6 years of additional obligated service. Since the 10% constraint is no longer current and the Marine Corps only offers four-year reenlistments, the more complicated formulation is not necessary in this thesis.

## E. THESIS OUTLINE

In Chapter II, the SRB multiplier problem is formulated as a generalized assignment problem. The data and variables are described along with the methodology used to compute model coefficients like the training costs in the object function. This model is then implemented and solved in the General Algebraic Modeling System ("GAMS"; see Rosenthal 2007), with an Excel interface, for testing purposes. Although the GAMS implementation and solution is a perfectly reasonable one, it is unsatisfactory for the final product of this thesis, because it relies on an unfamiliar computational environment for manpower planners. Instead, this solution is used in Chapter III to check the accuracy of a solution method based on Lagrangian relaxation, whose implementation and testing in VBA and Excel is also described in Chapter III. Chapter IV presents conclusions and recommendations for further work. Appendices A and B contain the source code for the GAMS and Excel VBA models, respectively. Appendix C contains sample inputs and outputs from the Excel VBA model.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. GENERALIZED ASSIGNMENT MODEL

This chapter models the optimization of SRB multipliers across all MOSZs as a generalized assignment problem (GAP). First, the model terms and coefficients are defined. Second, the differences from DeWolfe's model and the current model are explained. Finally, the estimation methods used for some of the coefficients in the model are defined.

## A. PROBLEM FORMULATION

There is a total of *m* MOSZs that encompass all MOSs and Zones. Each MOSZ will take on one of *n* multiplier "levels," with level $j = 1$ corresponding to a multiplier value of $v_1 = 0$. Current SRB policy uses multipliers in increments of 0.5, up to a maximum of 5.0, for a total of 11 multiplier levels. Thus, each increase in level *j* results in an increase of 0.5 in $v_j$. For example, level $j = 2$ corresponds to $v_2 = 0.5$, $j = 3$ to $v_3 = 1.0$, and the maximum level of $j = 11$ corresponds to $v_{11} = 5$. The model is not limited to a predefined number of multiplier levels, as long as $v_{j+1} - v_j$ is constant for all levels $j$.

Currently, there is no set maximum multiplier value assigned to any single MOSZ or group of MOSZs that is different from the overall maximum multiplier mentioned above. However, the multiplier cannot exceed a value that would result in a bonus that exceeds the maximum allowable bonus. The maximum allowable multiplier is easy to compute and is represented through $n_i$, the maximum multiplier level for MOSZ $i$.

Each MOSZ has an associated cost (budget consumption) and a penalty (objective-function value) that depends on the expected number of reenlistments for the chosen multiplier. The overall objective is to assign each MOSZ a multiplier that minimizes the total penalty and keeps the total cost under budget. The following

formulation of the SRB multiplier problem as a GAP is similar to DeWolfe's but some variables and parameters have been changed to account for Lat Moves and the use of non-integer multipliers.

**1. Indices**

$i = 1, ..., m$ MOSZ

$j = 1, ..., n_i$ allowable SRB multiplier levels for MOS $i$

**2. Parameters [units]**

$d_{ij}$ reenlistment-target deviation penalty for MOSZ $i$ with multiplier level $j$ [arbitrary penalty units]

$c_{ij}$ budget outlay incurred by setting multiplier level to $j$ for MOSZ $i$ [dollars]

$B$ budget [dollars]

**3. Decision Variables**

$x_{ij}$ 1 if multiplier level $j$ is selected for MOSZ $i$, 0 otherwise

**4. Formulation for GAP**

$$z^* = \min \sum_{i=1}^{m} \sum_{j=1}^{n_i} d_{ij} x_{ij} \qquad \text{(P1)}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \sum_{j=1}^{n_i} c_{ij} x_{ij} \le B \qquad \text{(C1)}$$

$$\sum_{j=1}^{n_i} x_{ij} = 1 \qquad \forall i \qquad \text{(C2)}$$

$$x_{ij} \in \{0,1\} \; \forall i, j \qquad \text{(C3)}$$

The objective function (P1) sums penalties across all MOSZs. The penalty is 0 for a reenlistment target that is met exactly, so the best possible objective-function value is 0. The first constraint (C1) limits total bonus expenditures to the available budget (at least in terms of expected values). Note that $c_{i1} = 0$ for all MOSZs $i$, that is, no cost is

incurred if no bonus is offered. The second constraint (C2) ensures that each MOSZ receives exactly one multiplier. This model follows DeWolfe's exactly, except that coefficients $d_{ij}$ and $c_{ij}$ are computed differently, as described below:

$$d_{ij} = \frac{\frac{T_i}{MTC} W_i}{A_i} \times \left( \begin{array}{l} \max\left\{0, \left(D_i - LM_i LMF_i\right) - RRR_{ij} E_i\right\}^2 \\ \qquad + Q \times \max\left\{0, RRR_{ij} E_i - \left(D_i - LM_i LMF_i\right)\right\}^2 \end{array} \right) \qquad \text{(Eqn 2.1)}$$

$$c_{ij} = RRR_{ij} E_i P_i v_j Y \qquad \text{(Eqn 2.2)}$$

where coefficients are defined by

$D_i$ number of Marines needed to reenlist in MOSZ $i$ [Marines]
$W_i$ exogenous weighting factor for MOSZ $i$ [unitless]
$P_i$ average monthly pay for Marines in MOSZ $i$ [dollars]
$v_j$ numerical multiplier value for multiplier level $j$
$RRR_{ij}$ expected reenlistment fraction for Marines in MOSZ $i$ and multiplier level $j$ [fraction]
$B$ budget [dollars]
$Q$ relative penalty weight of overages to underages [fraction]
$Y$ average years of reenlistment [years]
$MTC$ maximum training cost for all MOSZs [dollars]
$LM_i$ number of MOSZ $i$ school seats open for Lateral Moves [school seats]
$LMF_i$ fraction of available school seats to use for Lateral Moves into MOSZ $i$ [fraction]
$T_i$ training cost of MOSZ $i$ [dollars]
$A_i$ total number of Marines in MOSZ $i$ [Marines]
$E_i$ number of Marines eligible for reenlistment in MOSZ $i$ [Marines]

In Eqn 2.1, a Lat-Move functionality is added by the term $LM_i LMF_i$. This term can reduce the number of reenlistments that is needed from the eligible group of Marines in each MOSZ. Each MOSZ has a predetermined number of MOS school seats, $LM_i$, that are open for Lat Moves. Depending on the MOSZ, the number of school seats can range from zero to the total number of Marines that are needed to reenlist. In some MOSZs, manpower planners will want to use all available school seats and set $LMF_i = 1.0$. In others, planners will want more control over the number of new Marines

that LatMove into the MOSZ and set $LMF_i < 1$. Manpower planners can control the number of LatMoves by assigning $LMF_i$, a fraction of the school seats to use for each MOSZ. Historically there are always enough Lat-Move eligible Marines to fill school seats in a given FY, but manpower planners will need to verify that this assumption is still valid.

There are two changes to Eqn 2.2 over DeWolfe's formulation. First, the total bonus amount for each MOSZ is used in the budget constraints to account for the use of lump-sum payments instead of the anniversary payments in use in 1986. Second, a new term $v_j$ is introduced to account for the use of non-integer SRB multipliers. It is a parameter for the SRB multiplier value that is used in calculating the expected cost of using an arbitrary multiplier $v_j$. With this new parameter, manpower planners will not be limited to the integer-only multipliers of 1986 or the multipliers increments of 0.5 in use today.

## B. TRAINING-COST DATA

A key term in the weighting function for the objective of the SRB model is the training-cost parameter, which weights the deviation penalty linearly with increasing training cost: If a shortfall of $X$ Marines must occur in some MOSZ, it is better for that to occur in an MOS where the "replacement cost," based on training costs, is lower. Unfortunately, the Marine Corps no longer tracks training-school costs, because multiple sources of funding makes this accounting task difficult. Currently, the Marine Corps can only supply the school costs for non-Marine Corps schools because the Marine Corps pays a known amount for each Marine to attend. In order to overcome the lack of actual cost data, a new training-cost estimate is developed here, based on Zone, length of school or schools attended, graduation rate, and an estimate of the student's daily pay. All MOSZ training-cost parameter estimates use a variation of the following equation:

Training-cost estimate =

(Training Days $\times$ Daily Pay) / Graduation Rate (Eqn 2.3)

### 1. Training-Cost Estimate for Zone A

Zone-A training costs are broken into two categories: Initial training MOSs and Lat-Move-only MOSs. Both categories use Eqn 2.3 to estimate corresponding training costs but use different daily pay amounts to reflect the average pay received by a Marine while in training. This thesis applies, for initial training MOSs, the daily pay for an E-1 with less than two years in service; for Lat-Move-only MOSs, it uses the daily pay for an E-5 with more than four years in service.

### 2. Training–Cost Estimate for Zones B and C

Zone-B and -C training costs are also based on Eqn 2.3 but are more complicated because of the various tracks that a Marines' career can take in his MOS, an MOS that may, in fact, change. First, the Marine can be promoted and keep his original MOS. Under those circumstances, training cost is based on the MOS's Zone-A training cost. Second, a Marine can be promoted and receive a new MOS to distinguish new responsibilities. The new MOS can be supplied by a single MOS or by multiple "feeder MOSs." In the former case, the training-cost estimate is also based on the Zone-A training cost of the old MOS; otherwise the training cost is the average total training costs of the Zone-A feeder MOSs. In all of the above cases, it is possible that the Marine will be required to attend another MOS training school. If applicable, the new training cost for the school is estimated through Eqn 2.3 and is added to the prior-training cost estimate that is determined by the MOS track.

## C. GAMS MODEL

Using FY2004 data, the GAP is implemented and solved in GAMS using the XA solver (GAMS 2007). The FY2004 data set contains all 809 USMC MOSZs, of which 491 MOSZs need a multiplier greater than zero to reach their reenlistment goals. The final objective value is 57.8525 is obtained in 0.02 seconds of solver time, and is proven to be within 0.0006% of optimality. Of the 491 MOSZs, 385 receive a multiplier greater than zero. Table 2 shows the distribution of the eligible MOSZs and their selected multipliers.

Table 2. Distribution of SRB Multipliers in GAMS Solution

| Multiplier | 0.0 | 0.5 | 1.0 | 1.5 | 2.0 | 2.5 | 3.0 | 3.5 | 4.0 | 4.5 | 5.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MOSZ | 106 | 39 | 99 | 41 | 33 | 95 | 74 | 0 | 1 | 3 | 0 |

The eligible MOSZs have a reenlistment goal of 8,887 Marines, and with the chosen multipliers 5,620.75 Marines are expected to reenlist. The SRB budget for FY2004 is $54 million and only $474 goes unspent in the GAMS solution.

# III. SOLVING THE GENERALIZED ASSIGNMENT MODEL WITH LAGRANGIAN RELAXATION

Although the updated GAP formulation of the SRB multiplier problem solves quickly in GAMS, manpower planners cannot readily use this implementation due to lack of access to GAMS. In order to supply manpower planners with a model they can use, the GAP model is implemented and solved here using the Lagrangian-relaxation technique described by DeWolfe, but using a current programming language that can run in a computational environment that manpower planners have access to. The following section borrows heavily from DeWolfe (1986), and the reader should see that thesis for more details.

## A. LAGRANGIAN RELAXATION

The budget constraint (C1) in the GAP can be viewed as a "complicating constraint." That is, the model solves easily if (C1) is relaxed. Using Lagrangian relaxation, the budget constraint can be moved to the objective function using a Lagrangian multiplier that acts as a penalty for constraint violation, or as an incentive to spend. The relaxed formulation is:

For $\lambda \geq 0$,

$$\text{LR}(\lambda): \quad z(\lambda) = \min \sum_{i=1}^{m} \sum_{j=1}^{n_i} d_{ij} x_{ij} + \lambda \left( \sum_{i=1}^{m} \sum_{j=1}^{n_i} c_{ij} x_{ij} - B \right) \qquad \text{(P3)}$$

$$\text{s.t.} \quad \sum_{j=1}^{n_i} x_{ij} = 1 \qquad \forall i$$

$$x_{ij} \in \{0,1\} \; \forall i, j$$

The inner portion of P3 can be simplified and rewritten as:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n_i} (d_{ij} + \lambda c_{ij}) x_{ij} - \lambda B \qquad \text{(Eqn 3.1)}$$

Thus, for fixed $\lambda$, $z(\lambda)$ is computed by solving a simple selection problem: For each MOSZ $i$, choose $x_{ij}$ to be 1 for the largest value of $d_{ij} + \lambda c_{ij}$, and set other $x_{ij}$ to 0. It is well known, then, that $z(\lambda) \le z^*$, i.e., $z(\lambda)$ provides a lower bound on $z^*$.

The best lower bound is found by maximizing $z(\lambda)$ with respect to $\lambda$ to obtain $\lambda^*$. This can be done by bracketing the optimal $\lambda$ in some range $[L_l, L_u]$, and then narrowing the bracket through bisection search. $L_l = 0$ is always valid. For $\lambda$ sufficiently large, the solution to LR($\lambda$) will spend as few SRB budget dollars as possible. This will occur when setting $x_{1j} = 1$ for all $i$ is the optimal action, and this will occur when

$$d_{i1} + \lambda c_{i1} \le d_{ij} + \lambda c_{ij} \qquad \forall i, j \ge 2 \qquad \text{(Eqn 3.2)}$$

Since $c_{i1} = 0$, this leads to

$$d_{i1} \le d_{ij} + \lambda c_{ij} \qquad \forall i, j \ge 2. \qquad \text{(Eqn 3.3)}$$

Thus

$$L_u = \max\left\{(d_{i1} - d_{ij}) / c_{ij}\right\} \qquad \forall i, j \ge 2 \qquad \text{(Eqn 3.4)}$$

is a valid value for the right-hand side of the bracket.

The bisection search is based on the fact that if the budget constraint is violated for a given $\lambda$, then $\lambda^* \ge \lambda$; otherwise, $\lambda^* \le \lambda$.

The above methodology is implemented just as in DeWolfe (1986), but using VBA, and the best Lagrangian lower bound, $z(\lambda^*)$, is established. An upper bound on the optimal solution to the SRB multiplier problem is found by examining the feasible solution sets to $z(\lambda)$ that are encountered in the course of maximizing $z(\lambda)$. (When $\lambda > \lambda^*$, the solution to LR($\lambda$) is guaranteed to be feasible with respect to the budget constraint.)

The best solution identified while optimizing $z(\lambda)$ may not consume the entire budget. If unspent budget remains, it may be possible to improve the solution using the marginal-rate-of-return heuristic described by DeWolfe. The marginal rate of return for MOSZ $i$ is defined as:

$$ROR_i = (d_{ij} - d_{i,j+1})/(c_{i,j+1} - c_{ij}) \qquad \text{(Eqn 3.5)}$$

where the current feasible solution has $x_{ij} = 1$. The numerator in Eqn 3.5 calculates the amount $z(\lambda)$ can be improved by increasing the SRB level for MOSZ $i$ from its current level of $j$ to $j+1$; the denominator calculates the corresponding additional budget expenditure. In essence, $ROR_i$ calculates the "bang for the buck" for raising the multiplier level for each MOSZ individually. The heuristic repeatedly applies these rules until no additional budget can be feasibly consumed:

*a.* Given the current values for $x_{ij}$, compute $ROR_i$ for each MOSZ *i.*

b. Select MOSZ $i^*$ with the largest value of $ROR_i$ such that (a) setting $x_{ij} = 0$ and $x_{i,j+1} = 1$ does not violate the budget constraint, and (b) $j+1$ does not exceed the largest allowable SRB level for the MOSZ.

c. Set $x_{ij} = 0$ and $x_{i,j+1} = 1$.

## B. EXCEL VBA MODEL SOLUTION BY LAGRANGIAN RELAXATION

Using FY2004 data, the Lagrangian-based solution method for the GAP, as described above, is implemented in EXCEL using VBA. For the FY2004 data described previously, the final objective value of 57.8691 is obtained in 1.4 seconds, and is proven to be within 0.0018% of optimality. Of the 491 MOSZs eligible for a bonus, 385 receive a multiplier greater than zero. Table 2 shows the distribution of the eligible MOSZs and their selected multipliers.

Table 3. Distribution of SRB Multipliers in VBA Solution

| Multiplier | 0.0 | 0.5 | 1.0 | 1.5 | 2.0 | 2.5 | 3.0 | 3.5 | 4.0 | 4.5 | 5.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MOSZ | 106 | 39 | 94 | 42 | 35 | 97 | 74 | 0 | 1 | 3 | 0 |

The eligible MOSZs have a reenlistment goal of 8,887 Marines, and 5,621.51 Marines are expected to reenlist with the chosen multipliers. Of the $54 million budget, only $574 goes unspent.

## C. COMPARISON OF DIRECT AND LAGRANGIAN SOLUTIONS

The VBA and GAMS solution methods produce very similar outputs, with the VBA model having the higher objective-function value by 0.0166 units, or by 0.0287%. Of the 491 MOSZs that are eligible for an SRB, both models assign non-zero multipliers to the same 385 MOSZs. Of these MOSZs, the VBA solution assigns only 11 multiplier values that differ from the GAMS solution. Of these, nine multipliers increase by 0.5 in the VBA solution, one multiplier decreases by 0.5, and one multiplier increases by 1.5.

The largest difference in objective–function values results from the lone multiplier that decreases by 0.5. The decrease raises the objective value by 0.1057 units or 0.18% of the total VBA objective-function value. For the MOSZ with the largest multiplier difference in the VBA solution, the multiplier is raised from the GAMS-solution value of 1.5 to 3.0. Despite this large difference, the impact on the objective value is negligible: The objective value decreases by 0.1106 units or 0.06% of the total objective value. This small change in the objective value results from the fact that the relevant MOSZ contains only two Marines, and so even the worst multiplier assignment contributes little to the overall objective-function value (penalty).

# IV. CONCLUSIONS AND RECOMMENDATIONS

In summary, the model developed in this thesis provides Marine Corps manpower planners a new tool to help manage the Selective Reenlistment Bonus (SRB) program (USMC 1990). The model's solution specifies "SRB multipliers" for each Military Occupational Specialty (MOS) and Service Time Zone (Zone) combination that define the bonus amount, and are projected to encourage reenlistments that will (a) meet annual reenlistment targets as best possible, and (b) not exceed the available SRB budget. The new extended SRB optimization model and solution procedure incorporates key aspects of the methodology developed in DeWolfe (1986), but makes these improvements:

- Gives manpower planners the ability to exploit Lateral Moves,
- Allows a more flexible set of multiplier values that need not be integers,
- Makes budget and other parameters "user updateable," and
- Implements the model and solution procedure in a familiar computing environment for manpower planners, and at no cost (beyond software and hardware that these planners already possess).

The new model does have its limitations. When using the Lateral-Move functionality, the critical assumption about the availability of Marines eligible for "Lat Moves" needs to be checked. Also, the expected number of reenlistments for a given multiplier is based on a forecasting model that is assumed perfect.

Two areas that could improve the effectiveness of the model warrant further study:

- The development of a database that more accurately reflects true training costs, rather than the one used in this thesis, which estimates those costs based on training days and average pay,

- The expansion of the model to better forecast the expected cost of a given multiplier by breaking down the Marines in a MOSZ according to pay grade.

# LIST OF REFERENCES


Conatser, D. G., 2006. "Forecasting U.S. Marine Corps Reenlistments by MOS and Grade," M.S. Thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA, September.

DeWolfe, D.D., 1986. "Determination of Selective Reenlistment Bonus Multipliers in the United States Marine Corps," M.S. Thesis, Naval Postgraduate School, Monterey, CA, March.

DeWolfe, D.D., Stevens, J.G, and Wood, R.K., 1993. "Setting Military Reenlistment Bonuses," *Naval Research Logistics*, 40, 143–160.

GAMS, 2007. "XA." Retrieved August 15, 2007, Web site: http://www.gams.com/solvers/xa.pdf

Hattiangadi, A.U., Ackerman, D., Kimble, T., and Quester, A.O., 2004. "Cost-Benefit Analysis of Lump Sum Zone A, Zone B, and Zone C Reenlistment Study: Final Report," (CRM D0009652A2), The Center for Naval Analyses, Alexandria VA.

Manpower Plans and Policy Branch-Integration and Analysis, 2007. "WebManpower 101," presentation, Headquarters, Marine Corps, Manpower and Reserve Affairs, Quantico, VA.

Manpower Plans and Policy Branch-Integration and Analysis, 2007. "First Term Alignment Plan," presentation. Headquarters, Marine Corps, Manpower and Reserve Affairs, Quantico, VA.

Manpower Plans and Policy Branch-Integration and Analysis, 2007. "GAR Brief," presentation, Headquarters, Marine Corps, Manpower and Reserve Affairs, Quantico, VA.

Morgan, J.R., 2006. "Selective Reenlistment Bonus (SRB) Multiplier Assignment," Information Paper, Headquarters, Marine Corps, Manpower and Reserve Affairs, Quantico, VA.

Raymond, J.D., 2006. "Determining the Number of Reenlistments Necessary to Satisfy Future Force Requirements," M.S. Thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA, September.

Rosenthal, R.E., 2007. "GAMS-A User's Guide," Retrieved August 15, 2007, Web site: http://www.gams.com/docs/gams/GAMSUsersGuide.pdf

Training and Education Command, 2005. “Training Input Plan (TIP) FY 2005 Through FY 2009 – MOS Training Tracks.” Headquarters, Marine Corps, Marine Corps Combat Development Command, Quantico, VA.

USMC, 1990. Selective Reenlistment Bonus (SRB) Program (Marine Corps Order 7220.24M).

USMC, 1993. Enlisted Lateral Move Program (Marine Corps Order P1220.5J).

USMC, 2004. Enlisted Force Career Planning and Retention Manual (Marine Corps Order P1040.31J).

# APPENDIX A: GAMS SOURCE CODE

This appendix contains the source code for the nonlinear optimization problem solved in GAMS as a comparison and check for the Excel VBA solution.

```
$TITLE SRB Multiplier Optimization

*-----------DEFAULTS-------------------------------------------------

$OFFUPPER OFFSYMLIST OFFSYMXREF

OPTIONS
  LIMROW   =  0
  LIMCOL   =  0
  ITERLIM  =  500000
  RESLIM   =  100000
  SOLPRINT =  OFF
  DECIMALS =  2
  LP       =  XA
  RMIP     =  XA
  MIP      =  XA
  OPTCR    =  0.001
  OPTCA    =  0
  ;
*---------------------------------------------------------------------

$ONTEXT
      Original: 20 Jan 2007
            Author : Kent A. Robbins, Jr.
            Class :  Thesis
      Revised: 10 August 2007
             -Deleted Zone Loop (Now just associated with MOS)
             -Added read-in data function
             -Dropped Bonus limit constraint which is accounted for
                   in the data that is received from the Excel VBA
                   spreadsheet that writes the files.
             -Added fixing variables for Multipliers
             -Added output into Excel file
             -dropped OPTCA from 0.0 to .001
      Description:  SRB Multiplier Optimization Program that selects the
                   optimum multiplier to minimize the squared weighted
                   deviation of the differences between the expected
                   and needed number of reenlistments.  This program
                   is run in shell from Excel's VBA.
```

```
$OFFTEXT

*-----------INDICES          ---------------------------------------
SETS
   i  MOS

$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\MOSZoneList.dat

   m  Multiplier

$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\MultipleList.dat

;
*-----------DATA             ---------------------------------------

Scalar

 Q "relative weight of overages to shortages wrt reenlistment target"
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\Q.dat
 B "Total allowed budget"
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\Budget.dat
 Yrs "number of years for reenlistment"
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\Yrs.dat
 maxTrnCost "maximum training cost of all MOS's"
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\MaxTrnCost.dat
;

Parameter T(i)    "normalized cost of traing a Marine in MOS.Zone(i)"
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\TrnCost.dat
;

Parameter A(i)    "number of Marines in MOS.Zone(i)"
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\MOSPop.dat
;

Parameter E(i)    "number of Marines in MOS.Zone(i) eligible for reenlistment"
```

```
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\EASPop.dat
;

Parameter D(i)    "number of Marines required to reenlist in MOS.Zone(i)"
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\BoatSpace.dat
;

Parameter W(i)    "exogenous weighting factor for MOS.Zone(i)"
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\WgtFactor.dat
;

Parameter P(i)    "average monthly base bay of a Marine in MOS.Zone(i)"
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\AvgPay.dat
;

Parameter MultiplierNumber(m) "corresponding multiplier associated with set (m)"
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\MultipleNumbers.dat
;


Parameter Preset(i)      "gives the value of the preset multiplier for MOS.Zone(i)"
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\PresetMult.dat
;

TABLE
 RRR(i,m)    "reenlistment response rate for MOS.Zone(i) with Multiplier(m)"
$ONDELIM
$INCLUDE C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis Data Files\ReenlistmentRates.dat
$OFFDELIM
;
Parameter TEPDev(i,m)  "deviation of D(i)-RRR(i,m)*E(i) or 0 (target-expected)";
TEPDev(i,m)= max( 0, D(i)-RRR(i,m)*E(i) );

Parameter ETPDev(i,m)  "deviation of RRR(i,m)*E(i)-D(i) or 0 (expected-target)";
ETPDev(i,m)= max( 0, RRR(i,m)*E(i)-D(i) );

Parameter Wght(i)   "Weight function for MOS.Zone(i)";
Loop((i),  IF(A(i) > 0, Wght(i)= (T(i)/maxTrnCost)*W(i)/A(i);
```

```
        ELSE Wght(i)=0;););


*-----------VARIABLE----------------------------------------------------
BINARY VARIABLE
    y(i,m)   Binary var is 1 if Multiplier m is used for MOS.Zone(i)
    ;
VARIABLE
    Z      Total deviation for selection optimal multipliers
    ;
*-------------EQUATIONS-----------------------------------------------

EQUATIONS
     TOTALDEVIATION       total cost of SRB allocation
     PickOne(i)           only allow one SRB Multiplier
     TotalCost            total cost of picking mult m for MOS.Zone(i)

     ;
*-----------OBJECTIVE FUNCTION--------------------------------------

TOTALDEVIATION..
     Z =E= Sum((i,m),(Wght(i)*(TEPDev(i,m)**2+Q*ETPDev(i,m)**2))*y(i,m))
     ;

*-----------CONSTRAINTS---------------------------------------------

PickOne(i)..
     Sum(m, y(i,m)) =E= 1;
TotalCost..
     Sum((i,m), RRR(i,m)*E(i)*y(i,m)*MultiplierNumber(m)*P(i)*Yrs) =L= B;


*This loop will fix multipliers if the MOS.Zone(i) as a preset values.
LOOP((i)$(Preset(i)>-1),
 Loop((m), if(MultiplierNumber(m) = Preset(i), y.fx(i,m) = 1;););
);

MODEL SRB /ALL/ ;
SOLVE SRB USING MIP MINIMIZING Z ;

file out /C:\Documents and Settings\Kent Robbins\My Documents\Thesis\GAMS Thesis
Data Files\SRBOptResults.csv/;
put out;

*This loop writes the solution to a file that will be read in by Excel.
```

```
LOOP(i,
  LOOP(m$(y.l(i,m)=1),
     put MultiplierNumber(m),',';
  );
  put /;
);

putclose out;
;
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX B: VBA SOURCE CODE

This appendix contains the source code for the VBA modules that are used to set up the SRB optimization problem for both the GAMS and VBA models. This appendix also contains the module that solves the Lagrangian relaxation to the Generalized Assignment Model solved in GAMS.

```
'File:  Checks Module in VBA
'Date Created:  12 December 2006
'Last Updated:  15 August 2007
'
'Checks is a VBA Module that contains a list of subroutines that will perform checks on
'the data set to verify that it is complete in order to run the optimization software.
'
'@ Author Kent Robbins, Jr.
'
Option Explicit
Public lNumNodes As Long
Public lMaxIndex As Long
Public sNodeNames() As String
Public lIndex() As Long
Public lNumErrors As Long

'FillZoneABtSpaces() is a subroutine that will compute the correct number of
'reenlistments for Zone A that are needed when BNA school seats are used.
'
'@param - None
'
Sub FillZoneABtSpaces()
   Dim rZoneABtSpacesAvail As Range
   Dim rZoneABNASeatsAvail As Range
   Dim rZoneABNASeatUsage As Range
   Dim rZoneABtSpaceAdjust As Range
   Dim iLength As Integer, i As Integer
   Set rZoneABtSpacesAvail = wsMOSZoneA.Range("ZoneABtSpaceStart", _
                  wsMOSZoneA.Range("ZoneABtSpaceStart").End(xlDown))
   Set rZoneABNASeatsAvail = wsMOSZoneA.Range("ZoneABNASeatStart", _
                  wsMOSZoneA.Range("ZoneABNASeatStart").End(xlDown))
   Set rZoneABNASeatUsage = wsMOSZoneA.Range("ZoneABNASeatUsageStart", _
                  wsMOSZoneA.Range("ZoneABNASeatUsageStart").End(xlDown))
   iLength = rZoneABtSpacesAvail.Rows.Count
   Set rZoneABtSpaceAdjust = wsMOSZoneA.Range("ZoneABSAdjStart", _
                  wsMOSZoneA.Range("ZoneABSAdjStart").Offset(iLength, 0))
```

```
   For i = 1 To iLength
      rZoneABtSpaceAdjust(i) = Application.WorksheetFunction.Max _
                        ((rZoneABtSpacesAvail(i) - Application. _
                        WorksheetFunction.RoundUp(rZoneABNASeatsAvail(i) * _
                        rZoneABNASeatUsage(i), 0)), 0)
   Next i
End Sub

'MOSZoneCheck() is a subroutine that will verify that the MOS is listed in the Training
'Cost Worksheet.  This subroutine gathers the data and calls subroutine CheckMOS to
'verify that MOS is listed.  If there are MOSs that are not listed a window will pop up
'and warn the user.
'
'@param - None
'
Sub MOSZoneCheck()
   'Data Sheet Ranges
   Dim rZoneAMOS As Range
   Dim rZoneBMOS As Range
   Dim rZoneCMOS As Range
   Dim rZoneAMOSTrnVal As Range
   Dim rZoneBMOSTrnVal As Range
   Dim rZoneCMOSTrnVal As Range
   'Training Sheet Ranges
   Dim rTrainMOSAList As Range
   Dim rTrainMOSBCList As Range
   Dim rTrainMOSACostList As Range
   Dim rTrainMOSBCCostList As Range
   'Check & Fill in Zone A Ranges
   Set rZoneAMOS = wsMOSZoneA.Range("ZoneAMOSStart", wsMOSZoneA.Range _
            ("ZoneAMOSStart").End(xlDown))
   Set rZoneAMOSTrnVal = wsMOSZoneA.Range("ZoneATrnCstStart", _
                wsMOSZoneA.Range("ZoneATrnCstStart").End(xlDown))
   Set rTrainMOSAList = wsTrainCostZoneA.Range("TrnMOSZoneAListStart", _
                wsTrainCostZoneA.Range("TrnMOSZoneAListStart") _
                .End(xlDown))
   Set rTrainMOSACostList = wsTrainCostZoneA.Range("TrnMOSZoneACostStart", _
                   wsTrainCostZoneA.Range("TrnMOSZoneACostStart") _
                   .End(xlDown))
   Call CheckMOS(rZoneAMOS, rZoneAMOSTrnVal, rTrainMOSAList, _
            rTrainMOSACostList)
   If lNumErrors > 0 Then
      MsgBox ("Error: There are MOS(s) in Zone A not listed in the Training" _
         & " Cost Data-Zone A worksheet." & vbCrLf & "  See Data - " _
         & "Zone A Workseet.")
```

```
    End If
    'Check & Fill in Zone B Ranges
    Set rZoneBMOS = wsMOSZoneB.Range("ZoneBMOSStart", wsMOSZoneB.Range _
                ("ZoneBMOSStart").End(xlDown))
    Set rZoneBMOSTrnVal = wsMOSZoneB.Range("ZoneBTrnCstStart", _
                    wsMOSZoneB.Range("ZoneBTrnCstStart").End(xlDown))
    Set rTrainMOSBCList = wsTrainCostZoneBC.Range("TrnMOSZoneBCListStart", _
                    wsTrainCostZoneBC.Range("TrnMOSZoneBCListStart") _
                    .End(xlDown))
    Set rTrainMOSBCCostList = wsTrainCostZoneBC.Range _
                        ("TrnMOSZoneBCCostStart", _
                        wsTrainCostZoneBC.Range _
                        ("TrnMOSZoneBCCostStart").End(xlDown))
    Call CheckMOS(rZoneBMOS, rZoneBMOSTrnVal, rTrainMOSBCList, _
              rTrainMOSBCCostList)
    If lNumErrors > 0 Then
       MsgBox ("Error: There are MOS(s) in Zone B not listed in the Training" _
            & " Cost Data-Zone BC worksheet." & vbCrLf & "  See Data - " _
            & "Zone B Workseet.")
    End If
    'Check & Fill in Zone C Ranges
    Set rZoneCMOS = wsMOSZoneC.Range("ZoneCMOSStart", wsMOSZoneC.Range _
                ("ZoneCMOSStart").End(xlDown))
    Set rZoneCMOSTrnVal = wsMOSZoneC.Range("ZoneCTrnCstStart", _
                    wsMOSZoneC.Range("ZoneCTrnCstStart").End(xlDown))
    Call CheckMOS(rZoneCMOS, rZoneCMOSTrnVal, rTrainMOSBCList, _
              rTrainMOSBCCostList)
    If lNumErrors > 0 Then
       MsgBox ("Error: There are MOS(s) in Zone C not listed in the Training" _
            & " Cost Data-Zone BC worksheet." & vbCrLf & "  See Data - " _
            & "Zone C Workseet.")
    End If
End Sub

'CheckMOS() is a subroutine that will verify that the MOS is listed in the
'Training Cost Worksheet. If the MOS is listed the training cost will be listed
'in the training cost column, if not an error will be written.
'
'@param - rMOSToCheck is the range of MOSs in the worksheet that need to be verified
'      - rTrnValToFill is the range where training costs will be filled in the
'        worksheet
'      - rMOSList is the range of MOSs that are listed in the Traing Cost Worksheet
'      - rTrnCost is the range of training costs for MOSs listed in the Traing Cost
'        Worksheet
'
```

```
Sub CheckMOS(rMOSToCheck As Range, rTrnValToFill As Range, _
        rMOSList As Range, rTrnCost As Range)
   Dim lNumMOS As Long
   Dim lNumMOSToCheck As Long
   lNumErrors = 0
   lNumNodes = 0
   lNumMOS = rMOSList.Rows.Count
   lNumMOSToCheck = rMOSToCheck.Rows.Count
   lMaxIndex = 3 * lNumMOS
   ReDim lIndex(1 To lMaxIndex) As Long
   ReDim sNodeNames(0 To 2 * lNumMOS) As String
   Dim i As Integer
   For i = 1 To lNumMOS
      AddNode rMOSList(i)
   Next i
   rTrnValToFill.Clear
   For i = 1 To lNumMOSToCheck
      If DoesContain(rMOSToCheck(i)) Then
         rTrnValToFill(i) = rTrnCost(i)
      Else
         rTrnValToFill(i) = "Error, MOS not LISTED"
         lNumErrors = lNumErrors + 1
      End If
   Next i
   Call FormatTwoDecimal(rTrnValToFill)
End Sub

'ReenlistmentRateCheckByOccFld() is a subroutine that will verify that the OccField
'is listed in the Reenlistment Rate Worksheet for the correct zone.  This
'subroutine will call ReenlistmentrateCheckMaster and make it verify by OccField.
'
'@param - None
'
Sub ReenlistmentRateCheckByOccFld()
   Call ReenlistmentRateCheckMaster(True)
End Sub

'ReenlistmentRateCheckByMOS() is a subroutine that will verify that the MOS is listed
'in the Reenlistment Rate Worksheet for the correct zone.  This subroutine will call
'ReenlistmentrateCheckMaster and make it verify by MOS.
'
'@param - None
'
Sub ReenlistmentRateCheckByMOS()
   Call ReenlistmentRateCheckMaster(False)
```

```
End Sub

'ReenlistmentRateCheckMaster() is a subroutine that will verify that the MOS/OccField
'is listed in the Reenlistment Rate Worksheet for the correct zone.  This subroutine
'gathers the data to use and calls the subroutine CheckRRR() to verify that the
'MOS/OccField is listed.  If the MOS/OccField is listed the reenlistment rates and its
'corresponding multiplier will be listed in the reenlistment rate column, if not an
'error will be written and the user will be notified.
'
'@param - CheckByOccField is a boolean variable that will be True if the reenlistment
'        rates need to be verified by OccField or false if the rates need to be
'        verified by MOS
'
Sub ReenlistmentRateCheckMaster(bCheckByOccField As Boolean)
    Dim lLength As Long
    Dim lWidth As Long
    'Data Sheet Ranges
    Dim rZoneAMOS As Range
    Dim rZoneAOccField As Range
    Dim rZoneAEASPop As Range
    Dim rZoneABSpace As Range
    Dim rZoneARRR As Range
    Dim rZoneBMOS As Range
    Dim rZoneBOccField As Range
    Dim rZoneBEASPop As Range
    Dim rZoneBBSpace As Range
    Dim rZoneBRRR As Range
    Dim rZoneCMOS As Range
    Dim rZoneCOccField As Range
    Dim rZoneCEASPop As Range
    Dim rZoneCBSpace As Range
    Dim rZoneCRRR As Range
    'Reenlistment Rates Ranges
    Dim rRRRZoneAOccField As Range
    Dim rRRRZoneBOccField As Range
    Dim rRRRZoneCOccField As Range
    Dim rRRRZoneA As Range
    Dim rRRRZoneB As Range
    Dim rRRRZoneC As Range
    'Check Zone A by OccField
    Set rZoneAMOS = wsMOSZoneA.Range("ZoneAMOSStart", wsMOSZoneA.Range _
            ("ZoneAMOSStart").End(xlDown))
    lLength = rZoneAMOS.Rows.Count
    Set rZoneAOccField = wsMOSZoneA.Range("ZoneAOccFieldStart", _
                wsMOSZoneA.Range("ZoneAOccFieldStart") _
```

```
                .Offset(lLength, 0))
Set rZoneAEASPop = wsMOSZoneA.Range("ZoneAEASPopStart", _
              wsMOSZoneA.Range("ZoneAEASPopStart") _
              .End(xlDown))
Set rZoneABSpace = wsMOSZoneA.Range("ZoneABSAdjStart", _
              wsMOSZoneA.Range("ZoneABSAdjStart") _
              .End(xlDown))
Set rRRRZoneAOccField = wsRRRZoneA.Range("RRRZoneAOccFieldStart", _
                 wsRRRZoneA.Range("RRRZoneAOccFieldStart") _
                 .End(xlDown))
Set rRRRZoneA = wsRRRZoneA.Range(wsRRRZoneA.Range("RRRZoneAStart") _
            .End(xlToRight), wsRRRZoneA.Range("RRRZoneAStart") _
            .End(xlDown))
lWidth = rRRRZoneA.Columns.Count
Set rZoneARRR = wsMOSZoneA.Range("ZoneARRRStart", wsMOSZoneA.Range _
            ("ZoneARRRStart").Offset(lLength + 1, lWidth + 1))
Call CheckRRR(rZoneAOccField, rZoneAMOS, rZoneAEASPop, rZoneABSpace, _
          rZoneARRR, rRRRZoneAOccField, rRRRZoneA, lLength, _
          lWidth, bCheckByOccField)
If lNumErrors > 0 Then
   MsgBox ("Error: There are OccField/MOS(s) in Zone A not listed in the " _
        & "Reenlistment Rates-Zone A worksheet." & vbCrLf & "  See Data - " _
        & "Zone A Workseet.")
End If
'Check Zone B by OccField
Set rZoneBMOS = wsMOSZoneB.Range("ZoneBMOSStart", wsMOSZoneB.Range _
            ("ZoneBMOSStart").End(xlDown))
lLength = rZoneBMOS.Rows.Count
Set rZoneBOccField = wsMOSZoneB.Range("ZoneBOccFieldStart", _
                wsMOSZoneB.Range("ZoneBOccFieldStart") _
                .Offset(lLength, 0))
Set rZoneBEASPop = wsMOSZoneB.Range("ZoneBEASPopStart", _
              wsMOSZoneB.Range("ZoneBEASPopStart") _
              .End(xlDown))
Set rZoneBBSpace = wsMOSZoneB.Range("ZoneBBSAdjStart", _
              wsMOSZoneB.Range("ZoneBBSAdjStart") _
              .End(xlDown))
Set rRRRZoneBOccField = wsRRRZoneB.Range("RRRZoneBOccFieldStart", _
                 wsRRRZoneB.Range("RRRZoneBOccFieldStart") _
                 .End(xlDown))
Set rRRRZoneB = wsRRRZoneB.Range(wsRRRZoneB.Range("RRRZoneBStart") _
            .End(xlToRight), wsRRRZoneB.Range("RRRZoneBStart") _
            .End(xlDown))
lWidth = rRRRZoneB.Columns.Count
Set rZoneBRRR = wsMOSZoneB.Range("ZoneBRRRStart", wsMOSZoneB.Range _
```

```
            ("ZoneBRRRStart").Offset(lLength + 1, lWidth + 1))
    Call CheckRRR(rZoneBOccField, rZoneBMOS, rZoneBEASPop, rZoneBBSpace, _
            rZoneBRRR, rRRRZoneBOccField, rRRRZoneB, lLength, _
            lWidth, bCheckByOccField)
    If lNumErrors > 0 Then
       MsgBox ("Error: There are OccField/MOS(s) in Zone B not listed in the " _
          & "Reenlistment Rates-Zone B worksheet." & vbCrLf & "  See Data - " _
          & "Zone B Workseet.")
    End If
    'Check Zone C by OccField
    Set rZoneCMOS = wsMOSZoneC.Range("ZoneCMOSStart", wsMOSZoneC.Range _
            ("ZoneCMOSStart").End(xlDown))
    lLength = rZoneCMOS.Rows.Count
    Set rZoneCOccField = wsMOSZoneC.Range("ZoneCOccFieldStart", _
              wsMOSZoneC.Range("ZoneCOccFieldStart") _
              .Offset(lLength, 0))
    Set rZoneCEASPop = wsMOSZoneC.Range("ZoneCEASPopStart", _
             wsMOSZoneC.Range("ZoneCEASPopStart") _
             .End(xlDown))
    Set rZoneCBSpace = wsMOSZoneC.Range("ZoneCBSAdjStart", _
             wsMOSZoneC.Range("ZoneCBSAdjStart") _
             .End(xlDown))
    Set rRRRZoneCOccField = wsRRRZoneC.Range("RRRZoneCOccFieldStart", _
                wsRRRZoneC.Range("RRRZoneCOccFieldStart") _
                .End(xlDown))
    Set rRRRZoneC = wsRRRZoneC.Range(wsRRRZoneC.Range("RRRZoneCStart") _
            .End(xlToRight), wsRRRZoneC.Range("RRRZoneCStart") _
            .End(xlDown))
    lWidth = rRRRZoneC.Columns.Count
    Set rZoneCRRR = wsMOSZoneC.Range("ZoneCRRRStart", wsMOSZoneC.Range _
            ("ZoneCRRRStart").Offset(lLength + 1, lWidth + 1))
    Call CheckRRR(rZoneCOccField, rZoneCMOS, rZoneCEASPop, rZoneCBSpace, _
            rZoneCRRR, rRRRZoneCOccField, rRRRZoneC, lLength, _
            lWidth, bCheckByOccField)
    If lNumErrors > 0 Then
       MsgBox ("Error: There are OccField/MOS(s) in Zone C not listed in the " _
          & "Reenlistment Rates-Zone C worksheet." & vbCrLf & "  See Data - " _
          & "Zone C Workseet.")
    End If
End Sub

'CheckRRR() is a subroutine that will verify that the MOS/OccField is listed in the
'Reenlistment Rate Worksheet for the correct zone.  If the MOS/OccField is listed
'the reenlistment rates and its corresponding multiplier will be listed in the
'reenlistment rate column,if not an error will be written.
```

```
'
'@param - rOccFieldZoneList is the range where the correct OccField will be listed
'        on the Data Sheet
'      - rZoneMOSList is the range of MOS's that need to be verified
'      - rEASPop is the range of EAS Populations corresponding to the MOS that is
'        to be verified
'      - rBSpace is the range of Boat Spaces available for the MOS that is to be
'        verified
'      - rZoneRRR is the range that will be filled with reenlistment rates from
'        the Reenlistment Rate Worksheet
'      - rRRRZoneList is the range of MOS or OccFields that the ZoneMOSList will be
'        verified against
'      - rRRRZone is the range of reenlistment rates corresponding to the RRRZoneList
'      - lNumToCheck is the number of MOS/OccFields to check
'      - lNumMult is the number of SRB multipliers that are being used
'      - bCheckByOccField is a boolean variable that will be True if the reenlistment
'        rates need to be verified by OccField or false if the rates need to be
'        verified by MOS
'
Sub CheckRRR(rOccFieldZoneList As Range, rZoneMOSList As Range, _
          rEASPop As Range, rBSpace As Range, rZoneRRR As Range, _
          rRRRZoneList As Range, rRRRZone As Range, lNumToCheck As Long, _
          lNumMult As Long, bCheckByOccField As Boolean)
   Dim lNumRRR As Long
   Dim lPoint As Long
   Dim sOccField As String
   Dim bContain As Boolean
   rOccFieldZoneList.Clear
   rZoneRRR.Clear
   lNumErrors = 0
   lNumNodes = 0
   lNumRRR = rRRRZoneList.Rows.Count
   lMaxIndex = 3 * lNumRRR
   ReDim lIndex(1 To lMaxIndex) As Long
   ReDim sNodeNames(0 To 2 * lNumRRR) As String
   Dim i As Integer
   Dim j As Integer
   Dim lRRRPointer() As Long
   ReDim lRRRPointer(1 To lMaxIndex)
   For i = 1 To lNumRRR
      AddNode rRRRZoneList(i)
      lRRRPointer(LookupNode(rRRRZoneList(i))) = i + 1
   Next i
   'Formating here so the 0's on the MOSs will show up
   Call FormatOccFieldText(rOccFieldZoneList)
```

```
    'Prints the multiplier Numbers
    For i = 1 To lNumMult
       rZoneRRR(1, i) = rRRRZone(1, i)
    Next i
    For i = 1 To lNumToCheck
       sOccField = Mid(rZoneMOSList(i), 1, 2)
       If bCheckByOccField Then
          bContain = DoesContain(sOccField)
          If bContain Then
             lPoint = lRRRPointer(LookupNode(sOccField))
          End If
       Else
          bContain = DoesContain(rZoneMOSList(i))

          If bContain Then
             lPoint = lRRRPointer(LookupNode(rZoneMOSList(i)))
          End If
       End If
       rOccFieldZoneList(i) = sOccField
       If bContain Then
          For j = 1 To lNumMult
             rZoneRRR(i + 1, j) = Application.WorksheetFunction.Round _
                         (rRRRZone(lPoint, j), 4)
          Next j
       Else
          If rEASPop(i) = 0 Then
             rZoneRRR(i + 1, 1) = "Error, MOS not LISTED and NOT needed due " _
                         & "to Zero EAS Population"
          ElseIf rBSpace(i) = 0 Then
             rZoneRRR(i + 1, 1) = "Error, MOS not LISTED and NOT needed due " _
                         & "to Zero Boat Spaces"
          Else
             rZoneRRR(i + 1, 1) = "Error, MOS not LISTED and NEEDED"
             lNumErrors = lNumErrors + 1
          End If
       End If
    Next i
    Call FormatTwoDecimal(rZoneRRR)
End Sub

'File:  HashCode Module in VBA
'Date Created:  March 2007
'Last Updated:  May 2007
'
'Checks is a VBA Module that contains a list of subroutines that will perform
```

```
'checks on the data set to verify that it is complete in order to run the
'optimization software.
'
'@ Author Matthew Caryle
'   Updated by Kent Robbins, Jr.
'
Option Explicit
'HashName is a function that will take a string variable and convert it to a
'numerical value for use in a hash code.
'
'@param - sName is a String Variable that will be converted to a numerical value
'@return - HashName is a Long variable that is converted from sName
'
Function HashName(sName As String) As Long
   'Converts an input string sName into a long integer between 1 and lMaxIndex
   Dim lValue As Long
   Dim i As Integer
   lValue = 0
   For i = 1 To Len(sName)
      lValue = (10 * lValue + Asc(Mid(sName, i, 1))) Mod lMaxIndex
   Next i
   HashName = lValue + 1
End Function

'AddNode is a subroutine that will add a variable to the hash code.
'
'@param - sName is a string variable that will be added to the hash code.
'
Sub AddNode(sName As String)
   Dim hv As Long
   hv = HashName(sName)
   Do While lIndex(hv) <> 0 And sNodeNames(lIndex(hv)) <> sName
      hv = hv + 1
      If hv > lMaxIndex Then
         hv = 1
      End If
   Loop
   If lIndex(hv) = 0 Then
      lNumNodes = lNumNodes + 1
      lIndex(hv) = lNumNodes
      sNodeNames(lNumNodes) = sName
   End If
End Sub

'LoopupNode is a function that will return the node number of the parameter.
```

```
'
'@param - sName is a string variable used to look up its associated node number.
'@return - LoopUpNode
'
Function LookupNode(sName As String) As Long
   Dim hv As Long
   hv = HashName(sName)
   Do While lIndex(hv) <> 0 And sNodeNames(lIndex(hv)) <> sName
      hv = hv + 1
      If hv > lMaxIndex Then
         hv = 1
      End If
   Loop
   LookupNode = lIndex(hv)
End Function

'DoesContain is a function that will compute the correct number of reenlistments for
'Zone A that are needed when BNA school seats are used.
'
'@param - sName is a string variable that needs to be check to see if it is in the
'        hash code.
'@return - DoesContain is a boolean variable.  True if sName is in the hash code,
'         otherwise false.
'
Function DoesContain(sName As String) As Boolean
   Dim hv As Long
   hv = HashName(sName)
   Do While lIndex(hv) <> 0 And sNodeNames(lIndex(hv)) <> sName
      hv = hv + 1
      If hv > lMaxIndex Then
         hv = 1
      End If
   Loop
   If lIndex(hv) = 0 Then
      DoesContain = False
   Else
      DoesContain = True
   End If
End Function

'File:  Reduction Module in VBA
'Date Created:  17 June 2007
'Last Updated:  25 July 2007
'
'Reduction is a VBA Modula that will combine all MOS from Zone A, B, and C onto a
```

```
'single worksheet containing the necessary data for the GAMS and VBA solver to find
'the Optimum SRB multiplier.  Reduction will only transfer MOSs that will not achieve
'the reenlistment goal without the use of a multiplier greater than 0.  Reduction
'will also check the cost of each multiplier and will assign it a 0.0 reenlistment rate
'if that multiplier violates the maximum bonus contstraint.
'
'@ Author Kent Robbins, Jr.
'
Option Explicit
'ReduceSizeMaster is a subroutine that will gather the data neccessary and call the
'function Reduce to execute the reduction and transfer of the MOSsin Zones A, B, and
'C that will not achieve the reenlistment goal with a multiplier of 0.  Results are
'transfered to the ReducedFull Problem Worksheet.
'
'@param - none
'
Sub ReduceSizeMaster()
    'Zone Data Ranges
    Dim rZoneAOccField As Range
    Dim rZoneAMOSField As Range
    Dim rZoneAZoneField As Range
    Dim rZoneAMOSPopField As Range
    Dim rZoneAEASPopField As Range
    Dim rZoneAPreMultField As Range
    Dim rZoneAWeightField As Range
    Dim rZoneAReupNeedField As Range
    Dim rZoneATrainCostField As Range
    Dim rZoneARRR As Range
    Dim dZoneAAvgPay As Double
    Dim rZoneBOccField As Range
    Dim rZoneBMOSField As Range
    Dim rZoneBZoneField As Range
    Dim rZoneBMOSPopField As Range
    Dim rZoneBEASPopField As Range
    Dim rZoneBPreMultField As Range
    Dim rZoneBWeightField As Range
    Dim rZoneBReupNeedField As Range
    Dim rZoneBTrainCostField As Range
    Dim rZoneBRRR As Range
    Dim dZoneBAvgPay As Double
    Dim rZoneCOccField As Range
    Dim rZoneCMOSField As Range
    Dim rZoneCZoneField As Range
    Dim rZoneCMOSPopField As Range
    Dim rZoneCEASPopField As Range
```

```
Dim rZoneCPreMultField As Range
Dim rZoneCWeightField As Range
Dim rZoneCReupNeedField As Range
Dim rZoneCTrainCostField As Range
Dim rZoneCRRR As Range
Dim dZoneCAvgPay As Double
Dim dMaxIndivBonus As Double
'OutPutZone Variables
Dim iMaxOutputLength As Integer
Dim iMaxOutputWidth As Integer
Dim rOutput As Range
Dim rAvgPays As Range
Dim i As Integer
Dim iLinePosition As Integer
dMaxIndivBonus = wsReducedProblem.Range("MaxIndBonus")
Set rZoneAOccField = wsMOSZoneA.Range("ZoneAOccFieldStart", _
                wsMOSZoneA.Range("ZoneAOccFieldStart") _
                .End(xlDown))
Set rZoneAMOSField = wsMOSZoneA.Range("ZoneAMOSStart", _
                wsMOSZoneA.Range("ZoneAMOSStart") _
                .End(xlDown))
Set rZoneAZoneField = wsMOSZoneA.Range("ZoneASRBZoneStart", _
                 wsMOSZoneA.Range("ZoneASRBZoneStart") _
                 .End(xlDown))
Set rZoneAMOSPopField = wsMOSZoneA.Range("ZoneAMOSPopStart", _
                  wsMOSZoneA.Range("ZoneAMOSPopStart") _
                  .End(xlDown))
Set rZoneAEASPopField = wsMOSZoneA.Range("ZoneAEASPopStart", _
                  wsMOSZoneA.Range("ZoneAEASPopStart") _
                  .End(xlDown))
Set rZoneAPreMultField = wsMOSZoneA.Range("ZoneAPresetStart", _
                   wsMOSZoneA.Range("ZoneAPresetStart") _
                   .End(xlDown))
Set rZoneAWeightField = wsMOSZoneA.Range("ZoneAWgtFctStart", _
                  wsMOSZoneA.Range("ZoneAWgtFctStart") _
                  .End(xlDown))
Set rZoneAReupNeedField = wsMOSZoneA.Range("ZoneABSAdjStart", _
                   wsMOSZoneA.Range("ZoneABSAdjStart") _
                   .End(xlDown))
Set rZoneATrainCostField = wsMOSZoneA.Range("ZoneATrnCstStart", _
                    wsMOSZoneA.Range("ZoneATrnCstStart") _
                    .End(xlDown))
Set rZoneARRR = wsMOSZoneA.Range(wsMOSZoneA.Range("ZoneARRRStart") _
            .End(xlToRight), wsMOSZoneA.Range("ZoneARRRStart") _
            .End(xlDown))
```

```
dZoneAAvgPay = wsMOSZoneA.Range("ZoneAAvgPay")
Set rZoneBOccField = wsMOSZoneB.Range("ZoneBOccFieldStart", _
                wsMOSZoneB.Range("ZoneBOccFieldStart") _
                .End(xlDown))
Set rZoneBMOSField = wsMOSZoneB.Range("ZoneBMOSStart", _
                wsMOSZoneB.Range("ZoneBMOSStart") _
                .End(xlDown))
Set rZoneBZoneField = wsMOSZoneB.Range("ZoneBSRBZoneStart", _
                 wsMOSZoneB.Range("ZoneBSRBZoneStart") _
                 .End(xlDown))
Set rZoneBMOSPopField = wsMOSZoneB.Range("ZoneBMOSPopStart", _
                   wsMOSZoneB.Range("ZoneBMOSPopStart") _
                   .End(xlDown))
Set rZoneBEASPopField = wsMOSZoneB.Range("ZoneBEASPopStart", _
                   wsMOSZoneB.Range("ZoneBEASPopStart") _
                   .End(xlDown))
Set rZoneBPreMultField = wsMOSZoneB.Range("ZoneBPresetStart", _
                    wsMOSZoneB.Range("ZoneBPresetStart") _
                    .End(xlDown))
Set rZoneBWeightField = wsMOSZoneB.Range("ZoneBWgtFctStart", _
                   wsMOSZoneB.Range("ZoneBWgtFctStart") _
                   .End(xlDown))
Set rZoneBReupNeedField = wsMOSZoneB.Range("ZoneBBSAdjStart", _
                     wsMOSZoneB.Range("ZoneBBSAdjStart") _
                     .End(xlDown))
Set rZoneBTrainCostField = wsMOSZoneB.Range("ZoneBTrnCstStart", _
                      wsMOSZoneB.Range("ZoneBTrnCstStart") _
                      .End(xlDown))
Set rZoneBRRR = wsMOSZoneB.Range(wsMOSZoneB.Range("ZoneBRRRStart") _
             .End(xlToRight), wsMOSZoneB.Range("ZoneBRRRStart") _
             .End(xlDown))
dZoneBAvgPay = wsMOSZoneB.Range("ZoneBAvgPay")
Set rZoneCOccField = wsMOSZoneC.Range("ZoneCOccFieldStart", _
                wsMOSZoneC.Range("ZoneCOccFieldStart") _
                .End(xlDown))
Set rZoneCMOSField = wsMOSZoneC.Range("ZoneCMOSStart", _
                wsMOSZoneC.Range("ZoneCMOSStart") _
                .End(xlDown))
Set rZoneCZoneField = wsMOSZoneC.Range("ZoneCSRBZoneStart", _
                 wsMOSZoneC.Range("ZoneCSRBZoneStart") _
                 .End(xlDown))
Set rZoneCMOSPopField = wsMOSZoneC.Range("ZoneCMOSPopStart", _
                   wsMOSZoneC.Range("ZoneCMOSPopStart") _
                   .End(xlDown))
Set rZoneCEASPopField = wsMOSZoneC.Range("ZoneCEASPopStart", _
```

```
                    wsMOSZoneC.Range("ZoneCEASPopStart") _
                    .End(xlDown))
Set rZoneCPreMultField = wsMOSZoneC.Range("ZoneCPresetStart", _
                     wsMOSZoneC.Range("ZoneCPresetStart") _
                     .End(xlDown))
Set rZoneCWeightField = wsMOSZoneC.Range("ZoneCWgtFctStart", _
                    wsMOSZoneC.Range("ZoneCWgtFctStart") _
                    .End(xlDown))
Set rZoneCReupNeedField = wsMOSZoneC.Range("ZoneCBSAdjStart", _
                     wsMOSZoneC.Range("ZoneCBSAdjStart") _
                     .End(xlDown))
Set rZoneCTrainCostField = wsMOSZoneC.Range("ZoneCTrnCstStart", _
                      wsMOSZoneC.Range("ZoneCTrnCstStart") _
                      .End(xlDown))
Set rZoneCRRR = wsMOSZoneC.Range(wsMOSZoneC.Range("ZoneCRRRStart") _
             .End(xlToRight), wsMOSZoneC.Range("ZoneCRRRStart") _
             .End(xlDown))
dZoneCAvgPay = wsMOSZoneC.Range("ZoneCAvgPay")
iMaxOutputLength = rZoneAOccField.Rows.Count + rZoneBOccField.Rows.Count _
               + rZoneCOccField.Rows.Count
iMaxOutputWidth = 9 + rZoneARRR.Columns.Count
iLinePosition = 2
Set rOutput = wsReducedProblem.Range("RFPOutputStart", _
           wsReducedProblem.Range("RFPOutputStart") _
           .Offset(iMaxOutputLength + 1, iMaxOutputWidth))
Set rAvgPays = wsReducedProblem.Range("AvgPayA", wsReducedProblem.Range _
            ("AvgPayA").Offset(3, 0))
rOutput.Clear
Call FormatOccFieldText(rOutput)
rOutput.HorizontalAlignment = xlCenter
rAvgPays.Clear
rAvgPays.HorizontalAlignment = xlCenter
rOutput(1, 1) = "Occ Field"
rOutput(1, 2) = "MOS"
rOutput(1, 3) = "Zone"
rOutput(1, 4) = "MOSPop"
rOutput(1, 5) = "EASPop"
rOutput(1, 6) = "PresetMult"
rOutput(1, 7) = "WgtFactor"
rOutput(1, 8) = "B/SPACE"
rOutput(1, 9) = "TrnCosts"
For i = 1 To rZoneARRR.Columns.Count
   rOutput(1, 9 + i) = rZoneARRR(1, i)
Next i
rAvgPays(1) = dZoneAAvgPay
```

```
    rAvgPays(2) = dZoneBAvgPay
    rAvgPays(3) = dZoneCAvgPay
    iLinePosition = Reduce(rZoneAOccField, rZoneAMOSField, rZoneAZoneField, _
                rZoneAMOSPopField, rZoneAEASPopField, rZoneAPreMultField, _
                rZoneAWeightField, rZoneAReupNeedField, rZoneATrainCostField, _
                rZoneARRR, rOutput, dZoneAAvgPay, dMaxIndivBonus, iLinePosition)
    iLinePosition = Reduce(rZoneBOccField, rZoneBMOSField, rZoneBZoneField, _
                rZoneBMOSPopField, rZoneBEASPopField, rZoneBPreMultField, _
                rZoneBWeightField, rZoneBReupNeedField, rZoneBTrainCostField, _
                rZoneBRRR, rOutput, dZoneBAvgPay, dMaxIndivBonus, iLinePosition)
    iLinePosition = Reduce(rZoneCOccField, rZoneCMOSField, rZoneCZoneField, _
                rZoneCMOSPopField, rZoneCEASPopField, rZoneCPreMultField, _
                rZoneCWeightField, rZoneCReupNeedField, rZoneCTrainCostField, _
                rZoneCRRR, rOutput, dZoneCAvgPay, dMaxIndivBonus, iLinePosition)
End Sub

'Reduce is a function that will execute the reduction and transfer of those MOSs in
'a Zone that will need a multiplier greater than 0 to possible achieve the reenlistment
'goal.  Reduce will also check the cost of using each multiplier and if the cost is
'greater than the maximum allowable bonus the reenlistment rate for that multiplier will
'be set to 0 so it will not be used.
'
'@param - rOccField is the range of OccFields of the Zones MOSs to be reduced.
'       - rMos is the range of MOSs in the Zone that is to reduced.
'       - rZone is a range variable and is the Zone that to be reduced for each MOS.
'       - rMosPop is a range variable of MOS populations for each MOS.
'       - rEASPop is a range variable of EAS populations for each MOS.
'       - rPreset is a range variable of preset multiplier value for each MOS.
'       - rWeight is a range variable of exogenous weighting factor for each MOS.
'       - rBSpace is a range variable of needed reenlistment numbers for each MOS.
'       - rTrnCst is a range variable of training costs for each MOS.
'       - rRRR is a range variable of reenlistment response rates for each multiplier.
'       - rOutput is a range where the MOS and data will be transfered to.
'       - dAvgPay is a double variable of the average pay of the Zone being reduced.
'       - dMaxBonus is a double variable of the maximum allowable bonus.
'       - iLine is an integer variable and a place holder for the location in the
'         output for the next Zone to start being transfered to.
'
'@return - Reduce is an integer variable and is the location marker for the output
'          for the next Zone that is to be reduced.
'
Function Reduce(rOccField As Range, rMos As Range, rZone As Range, _
            rMosPop As Range, rEASPop As Range, rPreset As Range, _
            rWeight As Range, rBSpace As Range, rTrnCst As Range, _
            rRRR As Range, rOutput As Range, dAvgPay As Double, _
```

```
        dMaxBonus As Double, iLine As Integer) As Integer
Dim dGap As Double
Dim i As Integer
Dim j As Integer
Dim bContinue As Boolean
dGap = rRRR(1, 2) - rRRR(1, 1)
For i = 1 To rOccField.Rows.Count
   If rEASPop(i) = 0 Then
      bContinue = False
   ElseIf rBSpace(i) = 0 Then
      bContinue = False
   Else
      bContinue = True
   End If
If bContinue Then
   If rPreset(i) > -1 Then
      rOutput(iLine, 1) = rOccField(i)
      rOutput(iLine, 2) = rMos(i)
      rOutput(iLine, 3) = rZone(i)
      rOutput(iLine, 4) = rMosPop(i)
      rOutput(iLine, 5) = rEASPop(i)
      rOutput(iLine, 6) = rPreset(i)
      rOutput(iLine, 7) = rWeight(i)
      rOutput(iLine, 8) = rBSpace(i)
      rOutput(iLine, 9) = rTrnCst(i)
      For j = 1 To rRRR.Columns.Count
         rOutput(iLine, 9 + j) = rRRR(i + 1, j)
      Next j
      iLine = iLine + 1
   ElseIf (rEASPop(i) * rRRR(i + 1, 1) < rBSpace(i)) Then
      rOutput(iLine, 1) = rOccField(i)
      rOutput(iLine, 2) = rMos(i)
      rOutput(iLine, 3) = rZone(i)
      rOutput(iLine, 4) = rMosPop(i)
      rOutput(iLine, 5) = rEASPop(i)
      rOutput(iLine, 6) = rPreset(i)
      rOutput(iLine, 7) = rWeight(i)
      rOutput(iLine, 8) = rBSpace(i)
      rOutput(iLine, 9) = rTrnCst(i)
      For j = 1 To rRRR.Columns.Count
         If (dGap * (j - 1) * dAvgPay * 4 > dMaxBonus) Then
            rOutput(iLine, 9 + j) = 0
         Else
            rOutput(iLine, 9 + j) = rRRR(i + 1, j)
         End If
```

```
            Next j
            iLine = iLine + 1
        End If
    End If
    Next i
    Reduce = iLine
End Function

'File:  SolveVBA Module in VBA
'Date Created:  17 June 2007
'Last Updated:  31 August 2007
'
'SolveVBA Module is a VBA module that will solve the Lagrangian relaxation problem of
'finging the optimum SRB multipliers to assign to each MOSZ based on the weighted
'penalty function occured with each multiplier and subject to certain buget
'limitations.
'
'@ Author Kent Robbins, Jr.
'
Option Explicit
Public dInfinity As Double
Public dBudget As Double
Public dTest As Double
Public iNumMOS As Integer
Public iNumMult As Integer
Public dGap As Double
Public dHLambda As Double
Public dZL As Double
Public dZU As Double
Public dZH As Double
Public dSumCost As Double
Public dFinalCost As Double
Public dTotalDeviation As Double
'SolveEXCEL is the master subroutine that run the VBA Model.  This subroutine will
'gather the pertinent data that is need in later subroutines to find the optimum
'SRB multipliers.
'
'@param - none
'
Sub SolveEXCEL()
    Dim dStartTime As Double
    dStartTime = Timer
    Dim dFinishTime As Double
    'Worksheet variables
    Dim rMOSData As Range
```

```
    Dim rBudgetInfo As Range
    Dim dCost() As Double
    Dim dDeviation() As Double
    Dim i As Integer
    Dim dSoln() As Double
    Dim dPreset() As Double
    Dim dOptimality As Double
    dTotalDeviation = 0#
    'Read in data from worksheets
    Set rMOSData = wsReducedProblem.Range(wsReducedProblem.Range _
                ("RFPOutputStart").End(xlToRight), wsReducedProblem.Range _
                ("RFPOutputStart").End(xlDown))
    Set rBudgetInfo = wsReducedProblem.Range("OverageWeight", _
                  wsReducedProblem.Range("OverageWeight").End(xlDown))
    dBudget = rBudgetInfo(2)
    dInfinity = 1 * (10 ^ 22)
    iNumMOS = rMOSData.Rows.Count
    iNumMult = rMOSData.Columns.Count - 9
    dGap = rMOSData(1, 11) - rMOSData(1, 10)
    ReDim dCost(1 To (iNumMOS - 1), 1 To iNumMult) As Double
    ReDim dDeviation(1 To (iNumMOS - 1), 1 To iNumMult) As Double
    ReDim dPreset(1 To (iNumMOS - 1), 1 To 2) As Double
    ReDim dSoln(1 To (iNumMOS - 1)) As Double
    For i = 2 To iNumMOS
       If rMOSData(i, 6) >= 0 Then
          dPreset(i - 1, 1) = rMOSData(i, 6)
          dPreset(i - 1, 2) = rMOSData(i, 6) / dGap + 1
       Else
          dPreset(i - 1, 1) = -1
          dPreset(i - 1, 2) = -1
       End If
    Next i
    Call FillCostandDeviation(rMOSData, rBudgetInfo, dCost(), dDeviation())
    Call Bound(dSoln(), dPreset(), dCost(), dDeviation())
    Call Heuris(dDeviation(), dCost(), dSoln(), dPreset())
    Call WriteResults(rMOSData, rBudgetInfo, dSoln())
    dFinishTime = Timer
    dOptimality = (dZH - dZL) / dZL * 100
    MsgBox ("VBA Solve has finished." & vbCrLf & "Solution is within " & Round _
          (dOptimality, 6) & "% of optimality" & vbCrLf & "Run time was " & _
          Round((dFinishTime - dStartTime), 3) & " seconds" & vbCrLf & "Object " _
          & "Function is " & Round(dTotalDeviation, 4) & vbCrLf & "Total Cost " _
          & "is $" & Round(dFinalCost * 1000, 2))
End Sub
'FillCostandDeviation is a subroutine that will fill the Cost and Deviation arrays
```

```
'for each MOSZ and multiplier combination.
'
'@param - rMOSData is a range variable that contains data for all the MOSZs.
'      - rBudgetInfo is a range variable that contains the budget and pay
'        information needed for the SRB Optimization problem.
'      - dCost is an array of doubles that will contain the cost for setting each
'        MOSZ with all available multipliers.
'      - dDeviation is an array of doubles that will contain the deviation for
'        setting each MOSZ with all available multipliers.
'
Sub FillCostandDeviation(rMOSData As Range, rBudgetInfo As Range, _
                    dCost() As Double, dDeviation() As Double)
   Dim i As Integer
   Dim dScaleFactor As Double
   Dim dWeight As Double
   Dim dEligible As Double
   Dim dNeededNum As Double
   Dim dTotalNum As Double
   Dim dTrainCost As Double
   Dim dAvgPay As Double
   Dim iContractLength As Integer
   Dim dMaxTrainCost As Double
   dScaleFactor = 1000#
   dMaxTrainCost = 0#
   dBudget = dBudget / dScaleFactor
   dTest = dInfinity / 1.1
   dHLambda = -dInfinity
   'Set max Training Cost
   For i = 2 To iNumMOS
      dMaxTrainCost = Application.WorksheetFunction.Max(dMaxTrainCost, _
                  rMOSData(i, 9))
   Next i
   For i = 2 To iNumMOS
      Dim dTotalWeight As Double
      Dim j As Integer
      Dim dDev As Double
      Dim dHL As Double
      dTotalNum = rMOSData(i, 4)
      dEligible = rMOSData(i, 5)
      dWeight = rMOSData(i, 7)
      dNeededNum = rMOSData(i, 8)
      dTrainCost = rMOSData(i, 9)
      dAvgPay = rBudgetInfo(rMOSData(i, 3) + 4)
      iContractLength = rBudgetInfo(3)
      If (dTrainCost < 1#) Then
```

```
            dTrainCost = 50#
        End If
        If (dTotalNum > 0#) Then
            dTotalWeight = dWeight * (dTrainCost / dMaxTrainCost) / dTotalNum
        Else
            dTotalWeight = dWeight * (dTrainCost / dMaxTrainCost)
        End If
        If (dEligible < 1#) Then
            dEligible = 1#
        End If
        For j = 1 To iNumMult
            dDev = dNeededNum - dEligible * rMOSData(i, 9 + j)
            If (dDev < 0#) Then
                dDeviation(i - 1, j) = dTotalWeight * rBudgetInfo(1) * (dDev ^ 2)
            Else
                dDeviation(i - 1, j) = dTotalWeight * (dDev ^ 2)
            End If
            If (rMOSData(i, 9 + j) = 0) Then
                dCost(i - 1, j) = dInfinity
            Else
                dCost(i - 1, j) = ((dEligible * rMOSData(i, 9 + j)) * dAvgPay * _
                            iContractLength * rMOSData(1, 9 + j)) / _
                            dScaleFactor
            End If
            If (j > 1) Then
                dHL = (dDeviation(i - 1, 1) - dDeviation(i - 1, j)) / dCost(i - 1, j)
                If (dHL > dHLambda) Then
                    dHLambda = dHL
                End If
            End If
        Next j
    Next i
End Sub

'Bound is a subroutine that will determine the upper and lower bounds of the
'Lagrangian relaxation and call two other subroutines to find feasible solution
'sets with given Lagrangian multipliers and find the best feasible solution set
'that has the lowest object function.
'
'@param - dSoln is an array of doubles that contains the multipliers solution set.
'       - dPreset is an array of doubles the list the preset multipliers if chosen.
'       - dCost is an array of doubles that will contain the cost for setting each
'         MOSZ with all available multipliers.
'       - dDeviation is an array of doubles that will contain the deviation for
'         settingeach MOSZ with all available multipliers.
```

```
'
Sub Bound(dSoln() As Double, dPreset() As Double, dCost() As Double, _
        dDeviation() As Double)
   Dim dEps As Double
   Dim dEndL As Double
   Dim dEndR As Double
   Dim dZUBest As Double
   Dim dAmbda As Double
   Dim dFLambda As Double
   Dim i As Integer
   dEps = dHLambda / 100000
   dEndL = 0#
   dEndR = 0.01 * dHLambda
   dZUBest = dInfinity
   i = 0
   Do
      dAmbda = (dEndL + dEndR) / 2#
      Call MinFcn(dAmbda, dPreset(), dCost(), dDeviation())
      If (dSumCost <= dBudget) Then
         dEndR = dAmbda
         If (dZU <= dZUBest) Then
            dZUBest = dZU
            dFLambda = dAmbda
         End If
         Exit Do
      End If
      dEndL = dAmbda
      dEndR = 10.01 * dEndR
      i = i + 1
      Loop Until (i = 2)
   Do
      dAmbda = (dEndL + dEndR) / 2#
      Call MinFcn(dAmbda, dPreset(), dCost(), dDeviation())
      If (dSumCost <= dBudget) Then
         dEndR = dAmbda
         If (dZU <= dZUBest) Then
            dZUBest = dZU
            dFLambda = dAmbda
         End If
      Else
         dEndL = dAmbda
      End If
   Loop Until ((dEndR - dEndL) <= dEps)
   Call MinFeasible(dFLambda, dSoln(), dPreset(), dDeviation(), dCost())
End Sub
```

```
'MinFcn is a subroutine that will determine the minimum objective function for
'the current Lagrangian multiplier.
'
'@param - dAmbda is the Lagrangian multiplier current used.
'      - dPreset is an array of doubles the list the preset multipliers if chosen.
'      - dCost is an array of doubles that will contain the cost for setting each
'        MOSZ with all available multipliers.
'      - dDeviation is an array of doubles that will contain the deviation for
'        setting each MOSZ with all available multipliers.
'
Sub MinFcn(dAmbda As Double, dPreset() As Double, dCost() As Double, _
        dDeviation() As Double)
   Dim dCelMin As Double
   Dim dCelTot As Double
   Dim dCelObj As Double
   Dim dCObj As Double
   Dim dZJ As Double
   Dim iIndex As Integer
   Dim i As Integer
   Dim j As Integer
   Dim bContinue As Boolean
   dSumCost = 0#
   dCelTot = 0#
   dZU = 0#
   For i = 1 To (iNumMOS - 1)
      bContinue = True
      Do
         If (dPreset(i, 1) > -1) Then
            dCelMin = dDeviation(i, dPreset(i, 2)) + dAmbda * dCost(i, _
                  dPreset(i, 2))
            dCObj = dDeviation(i, dPreset(i, 2))
            iIndex = dPreset(i, 2)
            Exit Do
         End If
         dCelMin = dInfinity
         iIndex = 0
         For j = 1 To (iNumMult)
            If (dDeviation(i, j) <= dTest) Then
               dZJ = dDeviation(i, j) + dAmbda * dCost(i, j)
               dCelObj = dDeviation(i, j)
               If (dZJ <= dCelMin) Then
                  dPreset(i, 2) = j
                  dCelMin = dZJ
                  dCObj = dCelObj
```

```
                    iIndex = j
                End If
            End If
        Next j
        bContinue = False
    Loop Until bContinue = False
    dCelTot = dCelTot + dCelMin
    dZU = dZU + dCObj
    dSumCost = dSumCost + dCost(i, iIndex)
  Next i
  dZL = dCelTot - dAmbda * dBudget
End Sub

'MinFeasible is a subroutine that will determine minimum feasible object value
'solution set with the final Langrangian multiplier.  This is the feasible solution
'set with the lowest objective value.
'
'@param - dFLambda is a double variable and is the final chosen Langrangian
'        multiplier.
'       - dSoln is an array of doubles that contains the multipliers solution set.
'       - dPreset is an array of doubles the list the preset multipliers if chosen.
'       - dCost is an array of doubles that will contain the cost for setting each
'        MOSZ with all available multipliers.
'       - dDeviation is an array of doubles that will contain the deviation for
'        setting each MOSZ with all available multipliers.
'
Sub MinFeasible(dFLambda As Double, dSoln() As Double, dPreset() As Double, _
            dDeviation() As Double, dCost() As Double)
  Dim dCelTot As Double
  Dim dCObj As Double
  Dim dCelMin As Double
  Dim dZJ As Double
  Dim dCelObj As Double
  Dim iIndex As Integer
  Dim i As Integer
  Dim j As Integer
  Dim bContinue As Boolean
  dSumCost = 0#
  dCelTot = 0#
  dZU = 0#
  For i = 1 To (iNumMOS - 1)
    Do
      If (dPreset(i, 1) > -1) Then
        dCObj = dDeviation(i, dPreset(i, 2))
        iIndex = dPreset(i, 2)
```

```
            Exit Do
         End If
         dCelMin = dInfinity
         iIndex = 0
         For j = 1 To (iNumMult)
            If (dDeviation(i, j) <= dTest) Then
               dZJ = dDeviation(i, j) + dFLambda * dCost(i, j)
               dCelObj = dDeviation(i, j)
               If (dZJ <= dCelMin) Then
                  dSoln(i) = j
                  dCelMin = dZJ
                  dCObj = dCelObj
                  iIndex = j
               End If
            End If
         Next j
         bContinue = False
      Loop Until bContinue = False
      dZU = dZU + dCObj
      dSumCost = dSumCost + dCost(i, iIndex)
   Next i
End Sub

'Heuris is a subroutine that will improve the solution to the Langranian relaxation
'of the SRB multiplier problem by using a heuristic to consume any residual budget.
'
'@param - dDeviation is an array of doubles that will contain the deviation for
'        setting each MOSZ with all available multipliers.
'      - dCost is an array of doubles that will contain the cost for setting each
'        MOSZ with all available multipliers.
'      - dSoln is an array of doubles that contains the multipliers solution set.
'      - dPreset is an array of doubles the list the preset multipliers if chosen.
'
Sub Heuris(dDeviation() As Double, dCost() As Double, dSoln() As Double, _
        dPreset() As Double)
   Dim dCost1 As Double
   Dim dCost2 As Double
   Dim i As Integer
   Dim bRoomInBudget As Boolean
   dCost1 = dSumCost
   bRoomInBudget = True
   While bRoomInBudget = True
      If dCost1 <= dBudget Then
         dCost2 = NewCost1(dDeviation(), dCost(), dSoln(), dPreset(), dCost1)
      End If
```

```
      If dCost2 = dCost1 Then
         bRoomInBudget = False
      Else
         dCost1 = dCost2
      End If
   Wend
   dZH = 0#
   dFinalCost = 0#
   For i = 1 To (iNumMOS - 1)
      dZH = dZH + dDeviation(i, dSoln(i))
      dFinalCost = dFinalCost + dCost(i, dSoln(i))
      dTotalDeviation = dTotalDeviation + dDeviation(i, dSoln(i))
   Next i
End Sub

'NewCost1 is a function used to calculate the rate of return for each MOSZ based on
'the rate of improvement to the objective function that can be achieve for each
'dollar.  Newcost1 will the pick the MOSZ with the largest ROR and increase the
'MOSZs solution byone step in the multiplier and calculating the new cost of the
'solution set.
'
'@param - dDeviation is an array of doubles that will contain the deviation for
'        setting each MOSZ with all available multipliers.
'      - dCost is an array of doubles that will contain the cost for setting each
'        MOSZ with all available multipliers.
'      - dSoln is an array of doubles that contains the multipliers solution set.
'      - dPreset is an array of doubles the list the preset multipliers if chosen.
'      - dOldCost1 is a double variable that is the current cost of the solution
'        set.
'
'@return - NewCost1 is a double variable that is the new cost of the new solution
'         set that has been heuristically imporved.
'
Function NewCost1(dDeviation() As Double, dCost() As Double, dSoln() As Double, _
            dPreset() As Double, dOldCost1 As Double) As Double
   Dim dUArray() As Double
   Dim dResid As Double
   Dim dRNum As Double
   Dim dRDen As Double
   Dim dUBest As Double
   Dim i As Integer
   Dim j As Integer
   Dim bContinueInner As Boolean
   Dim iIndex As Integer
   ReDim dUArray(1 To iNumMOS - 1)
```

```
dResid = dBudget - dOldCost1
For i = 1 To (iNumMOS - 1)
   Do 'continue do loop
      bContinueInner = True
      If (dPreset(i, 1) > -1) Then
         dUArray(i) = -dInfinity
         Exit Do 'exits continue do loop
      End If
      If (dSoln(i) >= iNumMult) Then
         dUArray(i) = -dInfinity
         Exit Do
      ElseIf (dDeviation(i, (dSoln(i) + 1)) > dTest) Then
         dUArray(i) = -dInfinity
         Exit Do
      End If
      dRNum = dDeviation(i, dSoln(i)) - dDeviation(i, (dSoln(i) + 1))
      dRDen = dCost(i, (dSoln(i) + 1)) - dCost(i, dSoln(i))
      If ((dRDen > dResid) Or (dRNum <= (0.1 * 10 ^ -8))) Then
         dUArray(i) = -dInfinity
         Exit Do 'exits continue do loop
      Else
         dUArray(i) = dRNum / dRDen
      End If
      If (dUArray(i) < 0#) Then
         dUArray(i) = -1 * dUArray(i)
      End If
   Loop Until bContinueInner = True 'end of continue loop
Next i
dUBest = -dInfinity
For i = 1 To (iNumMOS - 1)
   If (dUArray(i) > dUBest) Then
      dUBest = dUArray(i)
      iIndex = i
   End If
Next i
bContinueInner = True
Do 'bContinueInner do loop
   If (dUBest < (-dTest)) Then
      NewCost1 = dOldCost1
      Exit Do 'exits bContinueInner do loop
   End If
   NewCost1 = dOldCost1 - dCost(iIndex, dSoln(iIndex)) + dCost(iIndex, _
         (dSoln(iIndex) + 1))
   dSoln(iIndex) = dSoln(iIndex) + 1
Loop Until bContinueInner = True
```

```
End Function

'WriteResults is a subroutine that will write the final solution to the Opt Solution
'worksheet.
'
'@param - rMOSData is a range variable that contains data for all the MOSZs.
'       - rBudgetInfo is a range variable that contains the budget and pay
'         information needed for the SRB Optimization problem.
'       - dSoln is an array of doubles that contains the multipliers solution set.
'
Sub WriteResults(rMOSData As Range, rBudgetInfo As Range, dSoln() As Double)
    Dim i As Integer
    Dim rOutput As Range
    Set rOutput = wsOptSoln.Range("OutputStart", wsOptSoln.Range("OutputStart") _
            .Offset(iNumMOS, 10))
    rOutput.Clear
    rOutput(1, 1) = "Occ Field"
    rOutput(1, 2) = "MOS"
    rOutput(1, 3) = "Zone"
    rOutput(1, 4) = "EASPop"
    rOutput(1, 5) = "B/Space"
    rOutput(1, 6) = "Multiplier"
    rOutput(1, 7) = "ExpReupNumber"
    rOutput(1, 8) = "RndReUpNumber"
    rOutput(1, 9) = "Over/Under"
    rOutput(1, 10) = "Expected Cost"
    For i = 2 To iNumMOS
        rOutput(i, 1) = rMOSData(i, 1)
        rOutput(i, 2) = rMOSData(i, 2)
        rOutput(i, 3) = rMOSData(i, 3)
        rOutput(i, 4) = rMOSData(i, 5)
        rOutput(i, 5) = rMOSData(i, 8)
        rOutput(i, 6) = (dSoln(i - 1) - 1) * dGap
        rOutput(i, 7) = rMOSData(i, (rOutput(i, 6) / dGap + 1) + 9) * rOutput(i, 4)
        Call FormatTwoDecimal(rOutput(i, 7))
        rOutput(i, 8) = Application.WorksheetFunction.Round(rOutput(i, 7), 0)
        rOutput(i, 9) = rOutput(i, 8) - rOutput(i, 5)
        rOutput(i, 10) = rBudgetInfo(3, 1) * rBudgetInfo((rOutput(i, 3)) + 4, 1) * _
                rOutput(i, 6) * rOutput(i, 7)
        Call FormatTwoDecimal(rOutput(i, 10))
    Next i
End Sub

'File:  Utilities Module in VBA
'Date Created:  17 June 2007
```

```
'Last Updated:  25 July 2007
'
'Utilities is a Module that contains various formatting subroutines.
'
'@ Author Kent Robbins, Jr.
'
Option Explicit
'FormatTwoDecimal is a subroutine that will format a given range to two decimal
'places.
'
'@param - rTable is the range of values to be formatted.
'
Sub FormatTwoDecimal(rTable As Range)
   rTable.NumberFormat = "0.00"
End Sub

'FormatOccFieldText is a subroutine that will format a given range to text and
'align it to the right of the cell.
'
'@param - rTable is the range of cells to be formatted.
'
Sub FormatOccFieldText(rTable As Range)
   rTable.NumberFormat = "@"
   rTable.HorizontalAlignment = xlRight
End Sub

'File:  WriteFiles Module in VBA
'Date Created:  17 June 2007
'Last Updated:  25 July 2007
'
'WriteFiles is a VBA Modula that will write will read in the GAMS solutions and
'write the solutions along with information about the MOSs to the OptSoln worksheet.
'
'@ Author Kent Robbins, Jr.
'
Option Explicit
'WriteDataMaster is a subroutine that will gather information about MOSs that
'is needed by the GAMS solver and write that informatino into .dat files.
'@param - none
'
Sub WriteDataMaster()
   Dim i As Integer
   Dim j As Integer
   Dim rMOSData As Range
   Dim rAvgPays As Range
```

```
Dim rScalarData As Range
Dim sFileNames() As String
Dim sIndex() As String
Dim iLength As Integer
Dim dMaxTrnCost As Double
ChDrive Left$(ActiveWorkbook.Path, 1)
ChDir ActiveWorkbook.Path
Set rMOSData = wsReducedProblem.Range(wsReducedProblem.Range _
        ("RFPOutputStart").End(xlToRight), wsReducedProblem.Range _
        ("RFPOutputStart").End(xlDown))
Set rAvgPays = wsReducedProblem.Range("AvgPayA", wsReducedProblem _
        .Range("AvgPayA").Offset(3, 0))
Set rScalarData = wsReducedProblem.Range("OverageWeight", wsReducedProblem _
          .Range("OverageWeight").Offset(3, 0))
iLength = rMOSData.Rows.Count
dMaxTrnCost = 0#
ReDim sFileNames(1 To 15) As String
ReDim sIndex(1 To iLength) As String
sFileNames(1) = "MOSZoneList.dat"
sFileNames(2) = "MultiplierList.dat"
sFileNames(3) = "AvgPay.dat"
sFileNames(4) = "MOSPop.dat"
sFileNames(5) = "EASPop.dat"
sFileNames(6) = "PresetMult.dat"
sFileNames(7) = "WgtFactor.dat"
sFileNames(8) = "BoatSpace.dat"
sFileNames(9) = "TrnCost.dat"
sFileNames(10) = "ReenlistmentRates.dat"
sFileNames(11) = "MultiplierNumbers.dat"
sFileNames(12) = "Q.dat"
sFileNames(13) = "Budget.dat"
sFileNames(14) = "Yrs.dat"
sFileNames(15) = "MaxTrnCost.dat"
For i = 2 To iLength
   If (rMOSData(i, 3) = 1) Then
      sIndex(i) = """ + CStr(rMOSData(i, 2).Value) + "a" + """
   ElseIf (rMOSData(i, 3) = 2) Then
      sIndex(i) = """ + CStr(rMOSData(i, 2).Value) + "b" + """
   Else
      sIndex(i) = """ + CStr(rMOSData(i, 2).Value) + "c" + """
   End If
   If (rMOSData(i, 9) > dMaxTrnCost) Then
      dMaxTrnCost = rMOSData(i, 9)
   End If
Next i
```

```
   For i = 1 To 11
      Call WriteData(i, sFileNames(i), sIndex(), rMOSData, rAvgPays, iLength, _
            rScalarData)
   Next i
   For i = 12 To 15
      Dim dSend As Double
      If i = 15 Then
         dSend = dMaxTrnCost
      Else
         dSend = rScalarData(i - 11)
      End If
      Call WriteDataScalar(sFileNames(i), dSend)
   Next i
End Sub

'WriteDataScalar is a subroutine that will write scalar data into a .dat file for
'GAMS.
'
'@param - none
'
Sub WriteDataScalar(sNames As String, dData As Double)
   Dim iFN As Integer
   iFN = FreeFile()
   Open sNames For Output As iFN
   Print #iFN, "/" & dData & "/"
   Close iFN
End Sub

'WriteData is a subroutine that will write MOSZ data into a .dat file for
'GAMS.
'
'@param - iPosition As Integer is the index for the files to be written.
'      - sNames As String is the file name for the data to be written.
'      - sIndex() As String is the name of the MOSZ to be written.
'      - rMOSData As Range is the range of data to be printed.
'      - rPay As Range variable of the avg pays to be printed.
'      - iLength As Integer is the number of files to be printed.
'      - rScalarData As Range variable of all the scalar values needed.
'
Sub WriteData(iPosition As Integer, sNames As String, sIndex() As String, _
           rMOSData As Range, rPay As Range, iLength As Integer, _
           rScalarData As Range)
   Dim i As Integer, j As Integer, iFN As Integer, iNumMult As Integer
   iFN = FreeFile()
   Open sNames For Output As iFN
```

```
If (iPosition = 1) Then
   Print #iFN, "/"
   For i = 2 To iLength
      Print #iFN, sIndex(i);
      If i <> iLength Then
         Print #iFN, ",";
      End If
   Next i
   Print #iFN, "/"
ElseIf (iPosition = 2) Then
   Print #iFN, "/"
   For i = 1 To (rMOSData.Columns.Count - (9))
      Print #iFN, i;
      If i <> (rMOSData.Columns.Count - (9)) Then
         Print #iFN, ",";
      End If
   Next i
   Print #iFN, "/"
ElseIf (iPosition = 3) Then
   Print #iFN, "/"
   For i = 2 To iLength
      Print #iFN, sIndex(i) & "=";
      If (rMOSData(i, 3) = 1) Then
         Print #iFN, rPay(1)
      ElseIf (rMOSData(i, 3) = 2) Then
         Print #iFN, rPay(2)
      Else
         Print #iFN, rPay(3)
      End If
      If (i <> iLength) Then
         Print #iFN, ",";
      End If
   Next i
   Print #iFN, "/"
ElseIf (iPosition = 10) Then
   iNumMult = rMOSData.Columns.Count - 9
   Print #iFN, "dummy";
   For i = 1 To iNumMult
      Print #iFN, "," & i;
   Next i
   Print #iFN,
   For i = 2 To iLength
      Print #iFN, sIndex(i);
      For j = iPosition To rMOSData.Columns.Count
         Print #iFN, "," & rMOSData(i, j);
```

```
            Next j
            Print #iFN,
        Next i
    ElseIf (iPosition = 11) Then
        iNumMult = rMOSData.Columns.Count - 9
        Print #iFN, "/"
        For i = 1 To iNumMult
            Print #iFN, i & "=" & rMOSData(1, 9 + i);
            If (i <> iNumMult) Then
                Print #iFN, ","
            End If
        Next i
        Print #iFN, "/"
    ElseIf (iPosition = 12) Then
        Print #iFN, "/"
        For i = 1 To 4
            Print #iFN, i & "=" & rScalarData(i);
            If (i <> 4) Then
                Print #iFN, ","
            End If
        Next i
        Print #iFN, "/"
    Else
        Print #iFN, "/"
        For i = 2 To iLength
            Print #iFN, sIndex(i) & "=" & rMOSData(i, iPosition);
            If (i <> iLength) Then
                Print #iFN, ","
            End If
        Next i
        Print #iFN, "/"
    End If
    Close iFN
End Sub

'File:  Solve Module in VBA
'Date Created:  17 June 2007
'Last Updated:  30 July 2007
'
'SolveGAMS is a subroutine that will run the GAMS solution model in a shell
'program.
'
'@ Author Kent Robbins, Jr.
'
Option Explicit
```

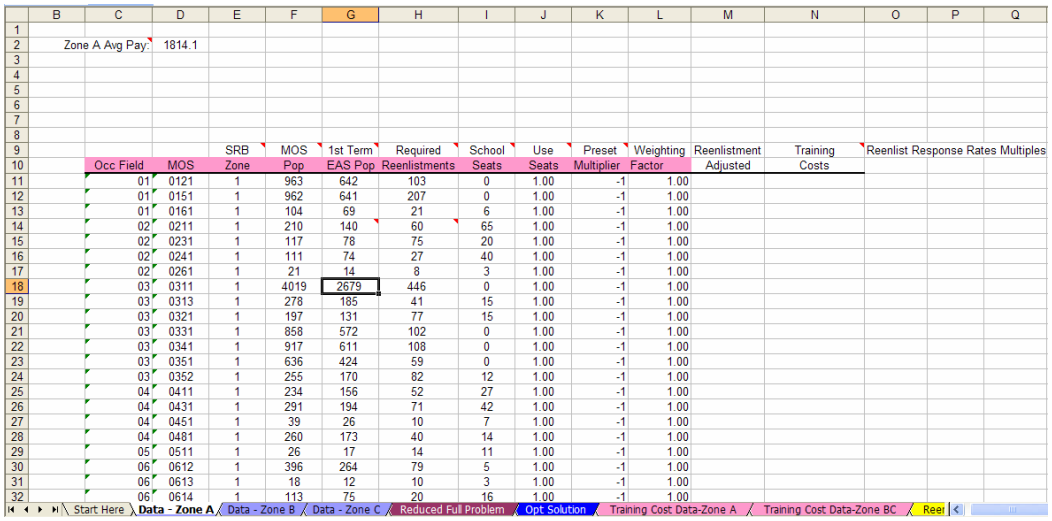
```
Sub SolveGAMS()
   ChDrive Left$(ActiveWorkbook.Path, 1)
   ChDir ActiveWorkbook.Path
   Shell "gams ThesisSRBModel.gms", vbNormalFocus
End Sub

'File:  ReadResults Module in VBA
'Date Created:  17 June 2007
'Last Updated:  14 July 2007
'
'ReadResults is a VBA Module that will read in the optimum SRB multiplier results from
'GAMS and print them to Opt Solution worksheet along with its associated data.
'
'@ Author Kent Robbins, Jr.
'
Option Explicit
'GAMSResults is a subroutine that will read in the optimum SRB multiplier results as
'an outputfrom GAMS.  Results will be outputted with other important data for the
'MOS's.
'
'@param - None
'
Sub GAMSResults()
   Dim iFN As Integer
   Dim i As Integer
   Dim j As Integer
   Dim iLength As Integer
   Dim r As Double
   Dim dGap As Double
   Dim dExpReUpNum As Double
   Dim rMOSData As Range
   Dim rOutput As Range
   Dim rAvgPays As Range
   ChDrive Left$(ActiveWorkbook.Path, 1)
   ChDir ActiveWorkbook.Path
   Set rMOSData = wsReducedProblem.Range(wsReducedProblem.Range _
            ("RFPOutputStart").End(xlToRight), wsReducedProblem.Range _
            ("RFPOutputStart").End(xlDown))
   Set rAvgPays = wsReducedProblem.Range("AvgPayA", wsReducedProblem _
            .Range("AvgPayA").Offset(3, 0))
   iLength = rMOSData.Rows.Count
   dGap = rMOSData(1, 11) - rMOSData(1, 10)
   Set rOutput = wsOptSoln.Range("OutputStart", wsOptSoln.Range("OutputStart") _
          .Offset(iLength, 10))
   rOutput(1, 1) = "Occ Field"
```

```
    rOutput(1, 2) = "MOS"
    rOutput(1, 3) = "Zone"
    rOutput(1, 4) = "EASPop"
    rOutput(1, 5) = "B/Space"
    rOutput(1, 6) = "Multiplier"
    rOutput(1, 7) = "ExpReupNumber"
    rOutput(1, 8) = "RndReUpNumber"
    rOutput(1, 9) = "Short/Over"
    rOutput(1, 10) = "Exact Expected Cost"
    iFN = FreeFile()
    Open "SRBOptResults.csv" For Input As iFN
    For i = 2 To iLength
        rOutput(i, 1) = rMOSData(i, 1)
        rOutput(i, 2) = rMOSData(i, 2)
        rOutput(i, 3) = rMOSData(i, 3)
        rOutput(i, 4) = rMOSData(i, 5)
        rOutput(i, 5) = rMOSData(i, 8)
        Input #iFN, r
        dExpReUpNum = rMOSData(i, (r / dGap + 1) + 9) * rOutput(i, 4)
        If dExpReUpNum = 0 Then
            rOutput(i, 6) = 0
            dExpReUpNum = rMOSData(i, 10) * rOutput(i, 4)
        Else
            rOutput(i, 6) = r
        End If
        rOutput(i, 7) = dExpReUpNum
        Call FormatTwoDecimal(rOutput(i, 7))
        rOutput(i, 8) = Application.WorksheetFunction.Round(rOutput(i, 7), 0)
        rOutput(i, 9) = rOutput(i, 8) - rOutput(i, 5)
        rOutput(i, 10) = 4 * rAvgPays(rOutput(i, 3)) * rOutput(i, 6) * rOutput(i, 7)
        Call FormatTwoDecimal(rOutput(i, 10))
        Input #iFN, r
    Next i
    Close iFN
End Sub
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX C: EXAMPLE INPUTS AND OUTPUTS

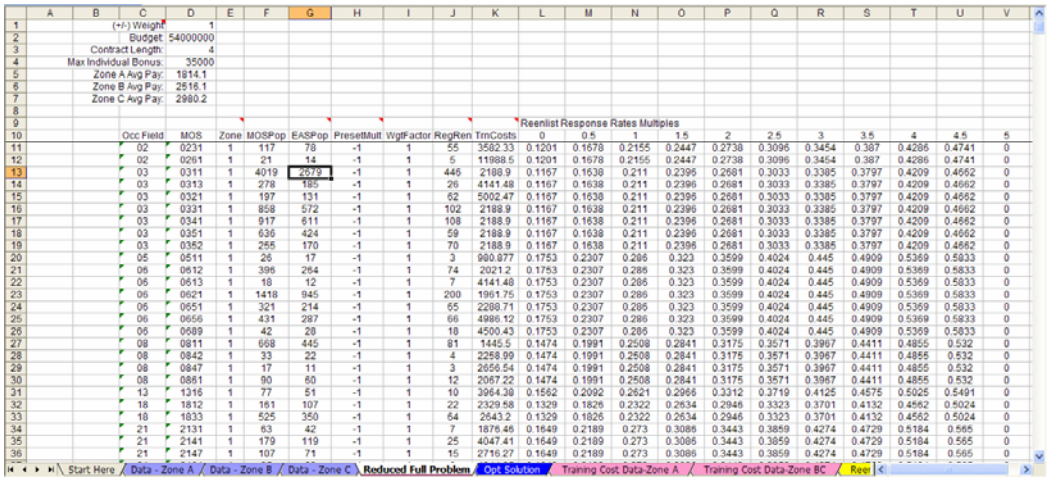
This appendix contains the examples of the inputs and outputs of the Excel interface that was used to solve the SRB multiplier selection problem for FY04.



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | Zone A Avg Pay: | 1814.1 | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | SRB | MOS | 1st Term | Required | School | Use | Preset | Weighting | Reenlistment | Training | Reenlist Response Rates Multiples | | |
| | Occ Field | MOS | Zone | Pop | EAS Pop | Reenlistments | Seats | Seats | Multiplier | Factor | Adjusted | Costs | | | |
| | 01 | 0121 | 1 | 963 | 642 | 103 | 0 | 1.00 | -1 | 1.00 | | | | | |
| | 01 | 0151 | 1 | 962 | 641 | 207 | 0 | 1.00 | -1 | 1.00 | | | | | |
| | 01 | 0161 | 1 | 104 | 69 | 21 | 6 | 1.00 | -1 | 1.00 | | | | | |
| | 02 | 0211 | 1 | 210 | 140 | 60 | 65 | 1.00 | -1 | 1.00 | | | | | |
| | 02 | 0231 | 1 | 117 | 78 | 75 | 20 | 1.00 | -1 | 1.00 | | | | | |
| | 02 | 0241 | 1 | 111 | 74 | 27 | 40 | 1.00 | -1 | 1.00 | | | | | |
| | 02 | 0261 | 1 | 21 | 14 | 8 | 3 | 1.00 | -1 | 1.00 | | | | | |
| | 03 | 0311 | 1 | 4019 | 2679 | 446 | 0 | 1.00 | -1 | 1.00 | | | | | |
| | 03 | 0313 | 1 | 278 | 185 | 41 | 15 | 1.00 | -1 | 1.00 | | | | | |
| | 03 | 0321 | 1 | 197 | 131 | 77 | 15 | 1.00 | -1 | 1.00 | | | | | |
| | 03 | 0331 | 1 | 858 | 572 | 102 | 0 | 1.00 | -1 | 1.00 | | | | | |
| | 03 | 0341 | 1 | 917 | 611 | 108 | 0 | 1.00 | -1 | 1.00 | | | | | |
| | 03 | 0351 | 1 | 636 | 424 | 59 | 0 | 1.00 | -1 | 1.00 | | | | | |
| | 03 | 0352 | 1 | 255 | 170 | 82 | 12 | 1.00 | -1 | 1.00 | | | | | |
| | 04 | 0411 | 1 | 234 | 156 | 52 | 27 | 1.00 | -1 | 1.00 | | | | | |
| | 04 | 0431 | 1 | 291 | 194 | 71 | 42 | 1.00 | -1 | 1.00 | | | | | |
| | 04 | 0451 | 1 | 39 | 26 | 10 | 7 | 1.00 | -1 | 1.00 | | | | | |
| | 04 | 0481 | 1 | 260 | 173 | 40 | 14 | 1.00 | -1 | 1.00 | | | | | |
| | 05 | 0511 | 1 | 26 | 17 | 14 | 11 | 1.00 | -1 | 1.00 | | | | | |
| | 06 | 0612 | 1 | 396 | 264 | 79 | 5 | 1.00 | -1 | 1.00 | | | | | |
| | 06 | 0613 | 1 | 18 | 12 | 10 | 3 | 1.00 | -1 | 1.00 | | | | | |
| | 06 | 0614 | 1 | 113 | 75 | 20 | 16 | 1.00 | -1 | 1.00 | | | | | |

Start Here | Data - Zone A | Data - Zone B | Data - Zone C | Reduced Full Problem | Opt Solution | Training Cost Data-Zone A | Training Cost Data-Zone BC | Reer

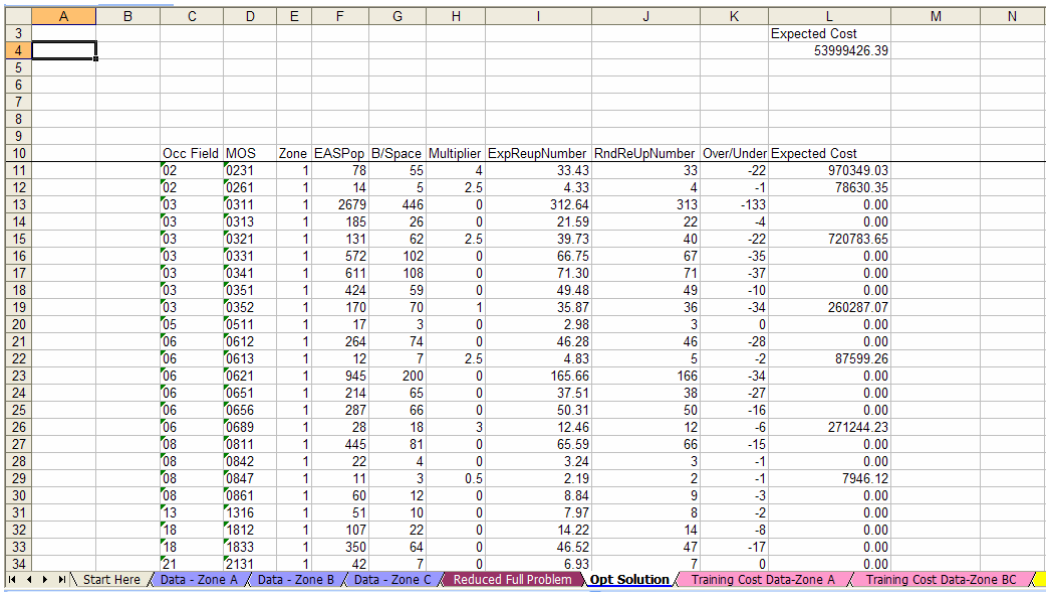
Figure 1. Excel Worksheet for Zone A Reenlistment Data.



| | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | (+/-) Weight: | 1 | | | | | | | | | | | | | | | | | | | | | | | |
| | | | Budget: | 54000000 | | | | | | | | | | | | | | | | | | | | | | | |
| | | | Contract Length: | 4 | | | | | | | | | | | | | | | | | | | | | | | |
| | | | Max Individual Bonus: | 35000 | | | | | | | | | | | | | | | | | | | | | | | |
| | | | Zone A Avg Pay: | 1814.1 | | | | | | | | | | | | | | | | | | | | | | | |
| | | | Zone B Avg Pay: | 2516.1 | | | | | | | | | | | | | | | | | | | | | | | |
| | | | Zone C Avg Pay: | 2980.2 | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | Reenlist Response Rates Multiples | | | | | | | | | | | | | | |
| | Occ Field | MOS | Zone | MOSPop | EASPop | PresetMult | WgtFactor | RegRen | TrmCosts | 0 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 | | | | | | | |
| | 02 | 0231 | 1 | 117 | 78 | -1 | 1 | 55 | 3582.33 | 0.1201 | 0.1678 | 0.2155 | 0.2447 | 0.2738 | 0.3096 | 0.3454 | 0.387 | 0.4286 | 0.4741 | 0 | | | | | | | |
| | 02 | 0261 | 1 | 21 | 14 | -1 | 1 | 5 | 11988.5 | 0.1201 | 0.1678 | 0.2155 | 0.2447 | 0.2738 | 0.3096 | 0.3454 | 0.387 | 0.4286 | 0.4741 | 0 | | | | | | | |
| | 03 | 0311 | 1 | 4019 | 2679 | -1 | 1 | 446 | 2188.9 | 0.1167 | 0.1638 | 0.211 | 0.2396 | 0.2681 | 0.3033 | 0.3385 | 0.3797 | 0.4209 | 0.4662 | 0 | | | | | | | |
| | 03 | 0313 | 1 | 278 | 185 | -1 | 1 | 26 | 4141.48 | 0.1167 | 0.1638 | 0.211 | 0.2396 | 0.2681 | 0.3033 | 0.3385 | 0.3797 | 0.4209 | 0.4662 | 0 | | | | | | | |
| | 03 | 0321 | 1 | 197 | 131 | -1 | 1 | 62 | 5002.47 | 0.1167 | 0.1638 | 0.211 | 0.2396 | 0.2681 | 0.3033 | 0.3385 | 0.3797 | 0.4209 | 0.4662 | 0 | | | | | | | |
| | 03 | 0331 | 1 | 858 | 572 | -1 | 1 | 102 | 2188.9 | 0.1167 | 0.1638 | 0.211 | 0.2396 | 0.2681 | 0.3033 | 0.3385 | 0.3797 | 0.4209 | 0.4662 | 0 | | | | | | | |
| | 03 | 0341 | 1 | 917 | 611 | -1 | 1 | 108 | 2188.9 | 0.1167 | 0.1638 | 0.211 | 0.2396 | 0.2681 | 0.3033 | 0.3385 | 0.3797 | 0.4209 | 0.4662 | 0 | | | | | | | |
| | 03 | 0351 | 1 | 636 | 424 | -1 | 1 | 59 | 2188.9 | 0.1167 | 0.1638 | 0.211 | 0.2396 | 0.2681 | 0.3033 | 0.3385 | 0.3797 | 0.4209 | 0.4662 | 0 | | | | | | | |
| | 03 | 0352 | 1 | 255 | 170 | -1 | 1 | 70 | 2188.9 | 0.1167 | 0.1638 | 0.211 | 0.2396 | 0.2681 | 0.3033 | 0.3385 | 0.3797 | 0.4209 | 0.4662 | 0 | | | | | | | |
| | 05 | 0511 | 1 | 26 | 17 | -1 | 1 | 3 | 980.877 | 0.1753 | 0.2307 | 0.286 | 0.323 | 0.3599 | 0.4024 | 0.445 | 0.4909 | 0.5369 | 0.5833 | 0 | | | | | | | |
| | 06 | 0612 | 1 | 396 | 264 | -1 | 1 | 74 | 2021.2 | 0.1753 | 0.2307 | 0.286 | 0.323 | 0.3599 | 0.4024 | 0.445 | 0.4909 | 0.5369 | 0.5833 | 0 | | | | | | | |
| | 06 | 0613 | 1 | 18 | 12 | -1 | 1 | 7 | 4141.48 | 0.1753 | 0.2307 | 0.286 | 0.323 | 0.3599 | 0.4024 | 0.445 | 0.4909 | 0.5369 | 0.5833 | 0 | | | | | | | |
| | 06 | 0621 | 1 | 1418 | 945 | -1 | 1 | 200 | 1961.75 | 0.1753 | 0.2307 | 0.286 | 0.323 | 0.3599 | 0.4024 | 0.445 | 0.4909 | 0.5369 | 0.5833 | 0 | | | | | | | |
| | 06 | 0651 | 1 | 321 | 214 | -1 | 1 | 65 | 2288.71 | 0.1753 | 0.2307 | 0.286 | 0.323 | 0.3599 | 0.4024 | 0.445 | 0.4909 | 0.5369 | 0.5833 | 0 | | | | | | | |
| | 06 | 0656 | 1 | 431 | 287 | -1 | 1 | 66 | 4986.12 | 0.1753 | 0.2307 | 0.286 | 0.323 | 0.3599 | 0.4024 | 0.445 | 0.4909 | 0.5369 | 0.5833 | 0 | | | | | | | |
| | 06 | 0689 | 1 | 42 | 28 | -1 | 1 | 18 | 4500.43 | 0.1753 | 0.2307 | 0.286 | 0.323 | 0.3599 | 0.4024 | 0.445 | 0.4909 | 0.5369 | 0.5833 | 0 | | | | | | | |
| | 08 | 0811 | 1 | 668 | 445 | -1 | 1 | 81 | 1445.5 | 0.1474 | 0.1991 | 0.2508 | 0.2841 | 0.3175 | 0.3571 | 0.3967 | 0.4411 | 0.4855 | 0.532 | 0 | | | | | | | |
| | 08 | 0842 | 1 | 33 | 22 | -1 | 1 | 4 | 2258.99 | 0.1474 | 0.1991 | 0.2508 | 0.2841 | 0.3175 | 0.3571 | 0.3967 | 0.4411 | 0.4855 | 0.532 | 0 | | | | | | | |
| | 08 | 0847 | 1 | 17 | 11 | -1 | 1 | 3 | 2656.54 | 0.1474 | 0.1991 | 0.2508 | 0.2841 | 0.3175 | 0.3571 | 0.3967 | 0.4411 | 0.4855 | 0.532 | 0 | | | | | | | |
| | 08 | 0861 | 1 | 90 | 60 | -1 | 1 | 12 | 2067.22 | 0.1474 | 0.1991 | 0.2508 | 0.2841 | 0.3175 | 0.3571 | 0.3967 | 0.4411 | 0.4855 | 0.532 | 0 | | | | | | | |
| | 13 | 1316 | 1 | 77 | 51 | -1 | 1 | 10 | 3964.38 | 0.1562 | 0.2092 | 0.2621 | 0.2966 | 0.3312 | 0.3719 | 0.4125 | 0.4575 | 0.5025 | 0.5491 | 0 | | | | | | | |
| | 18 | 1812 | 1 | 161 | 107 | -1 | 1 | 22 | 2329.58 | 0.1329 | 0.1826 | 0.2322 | 0.2634 | 0.2946 | 0.3323 | 0.3701 | 0.4132 | 0.4562 | 0.5024 | 0 | | | | | | | |
| | 18 | 1833 | 1 | 525 | 350 | -1 | 1 | 64 | 2643.2 | 0.1329 | 0.1826 | 0.2322 | 0.2634 | 0.2946 | 0.3323 | 0.3701 | 0.4132 | 0.4562 | 0.5024 | 0 | | | | | | | |
| | 21 | 2131 | 1 | 63 | 42 | -1 | 1 | 7 | 1876.46 | 0.1649 | 0.2189 | 0.273 | 0.3086 | 0.3443 | 0.3859 | 0.4274 | 0.4729 | 0.5184 | 0.565 | 0 | | | | | | | |
| | 21 | 2141 | 1 | 179 | 119 | -1 | 1 | 25 | 4047.41 | 0.1649 | 0.2189 | 0.273 | 0.3086 | 0.3443 | 0.3859 | 0.4274 | 0.4729 | 0.5184 | 0.565 | 0 | | | | | | | |
| | 21 | 2147 | 1 | 107 | 71 | -1 | 1 | 15 | 2716.27 | 0.1649 | 0.2189 | 0.273 | 0.3086 | 0.3443 | 0.3859 | 0.4274 | 0.4729 | 0.5184 | 0.565 | 0 | | | | | | | |

Start Here | Data - Zone A | Data - Zone B | Data - Zone C | Reduced Full Problem | Opt Solution | Training Cost Data-Zone A | Training Cost Data-Zone BC | Reer

Figure 2. Excel Worksheet of complied MOSZs and data need for the execution of the Model.



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | | | | | | | | | | | | Expected Cost | | |
| 4 | | | | | | | | | | | | 53999426.39 | | |
| 5 | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | |
| 10 | | | Occ Field | MOS | Zone | EASPop | B/Space | Multiplier | ExpReupNumber | RndReUpNumber | Over/Under | Expected Cost | | |
| 11 | | | 02 | 0231 | 1 | 78 | 55 | 4 | 33.43 | 33 | -22 | 970349.03 | | |
| 12 | | | 02 | 0261 | 1 | 14 | 5 | 2.5 | 4.33 | 4 | -1 | 78630.35 | | |
| 13 | | | 03 | 0311 | 1 | 2679 | 446 | 0 | 312.64 | 313 | -133 | 0.00 | | |
| 14 | | | 03 | 0313 | 1 | 185 | 26 | 0 | 21.59 | 22 | -4 | 0.00 | | |
| 15 | | | 03 | 0321 | 1 | 131 | 62 | 2.5 | 39.73 | 40 | -22 | 720783.65 | | |
| 16 | | | 03 | 0331 | 1 | 572 | 102 | 0 | 66.75 | 67 | -35 | 0.00 | | |
| 17 | | | 03 | 0341 | 1 | 611 | 108 | 0 | 71.30 | 71 | -37 | 0.00 | | |
| 18 | | | 03 | 0351 | 1 | 424 | 59 | 0 | 49.48 | 49 | -10 | 0.00 | | |
| 19 | | | 03 | 0352 | 1 | 170 | 70 | 1 | 35.87 | 36 | -34 | 260287.07 | | |
| 20 | | | 05 | 0511 | 1 | 17 | 3 | 0 | 2.98 | 3 | 0 | 0.00 | | |
| 21 | | | 06 | 0612 | 1 | 264 | 74 | 0 | 46.28 | 46 | -28 | 0.00 | | |
| 22 | | | 06 | 0613 | 1 | 12 | 7 | 2.5 | 4.83 | 5 | -2 | 87599.26 | | |
| 23 | | | 06 | 0621 | 1 | 945 | 200 | 0 | 165.66 | 166 | -34 | 0.00 | | |
| 24 | | | 06 | 0651 | 1 | 214 | 65 | 0 | 37.51 | 38 | -27 | 0.00 | | |
| 25 | | | 06 | 0656 | 1 | 287 | 66 | 0 | 50.31 | 50 | -16 | 0.00 | | |
| 26 | | | 06 | 0689 | 1 | 28 | 18 | 3 | 12.46 | 12 | -6 | 271244.23 | | |
| 27 | | | 08 | 0811 | 1 | 445 | 81 | 0 | 65.59 | 66 | -15 | 0.00 | | |
| 28 | | | 08 | 0842 | 1 | 22 | 4 | 0 | 3.24 | 3 | -1 | 0.00 | | |
| 29 | | | 08 | 0847 | 1 | 11 | 3 | 0.5 | 2.19 | 2 | -1 | 7946.12 | | |
| 30 | | | 08 | 0861 | 1 | 60 | 12 | 0 | 8.84 | 9 | -3 | 0.00 | | |
| 31 | | | 13 | 1316 | 1 | 51 | 10 | 0 | 7.97 | 8 | -2 | 0.00 | | |
| 32 | | | 18 | 1812 | 1 | 107 | 22 | 0 | 14.22 | 14 | -8 | 0.00 | | |
| 33 | | | 18 | 1833 | 1 | 350 | 64 | 0 | 46.52 | 47 | -17 | 0.00 | | |
| 34 | | | 21 | 2131 | 1 | 42 | 7 | 0 | 6.93 | 7 | 0 | 0.00 | | |

Start Here | Data - Zone A | Data - Zone B | Data - Zone C | Reduced Full Problem | **Opt Solution** | Training Cost Data-Zone A | Training Cost Data-Zone BC

Figure 3. Excel Worksheet with MOSZ data, multiplier solution and expected cost.

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California

4. Director, Training and Education MCCDC, Code C46
   Quantico, Virginia

5. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, Virginia

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

7. Director, Studies and Analysis Division, MCCDC, Code C45
   Quantico, Virginia

8. Director, Manpower Plans and Policy
   Manpower and Reserve Affairs, Code MPP-50
   Quantico, Virginia

9. Professor R. Kevin Wood
   Department of Operations Research
   Naval Postgraduate School
   Monterey, California

10. Professor Ronald D. Fricker, Jr.
    Department of Operations Research
    Naval Postgraduate School
    Monterey, California