A TAXONOMY OF ADVANCED LINEAR PROGRAMMING TECHNIQUES AND THE THEATER ATTACK MODEL

THESIS

Jack A. Jackson Jr.
Major, USAF

AFIT/GST/ENS/89-7

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY
Wright-Patterson Air Force Base, Ohio

AFIT/GST/ENS/89-7

A TAXONOMY OF ADVANCED LINEAR PROGRAMMING TECHNIQUES AND THE THEATER ATTACK MODEL

THESIS

Jack A. Jackson Jr.
Major, USAF

AFIT/GST/ENS/89-7

Approved for public release; distribution unlimited

AFIT/GST/ENS/89-7

A TAXONOMY OF ADVANCED LINEAR PROGRAMMING TECHNIQUES AND THE THEATER ATTACK MODEL

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science Operations Research

Jack A. Jackson Jr., M.S.

Major, USAF

March, 1989

Preface

This study considers advanced Linear Programming techniques and their usefulness for reducing the solution time for large-scale models; specifically the Theater Attack Model. The results suggest that time savings of 10–40% are not unreasonable, with the opportunity to reduce this specific model's CPU-time by 67%.

A number of agencies license products used within this study. MPS III is licensed to Ketron Incorporated, SCICONICS is licensed to SCICON Limited, MINOS 5.0 is licensed through Stanford University, KORBX is licensed through AT&T, and the X-System is copyrighted by Insight Incorporated. All of the above agencies are trademark names, as are IBM and ELXSI.

The list of analysts, practitioners, and fellow officers who assisted this research is long and distinguished. Without the help of Major Nick Reybrock and the AFCSA this project would have been impossible. Nick spent many hours on the phone answering "dumb" questions that helped narrow the focus. Without the service of Captain Mark Strovink and Mrs. Kris Larsen, both assigned to the AFIT computer division, again this project would have been grounded for lack of hardware, computer memory, and physical skill at manipulating a mainframe under stress. Without the support of MAC/AG, principally Major Jim Hill who arranged my training on the KORBX system and Captain Steve Wichmann who burned the late night oil for me, this analysis would certainly be incomplete. Without the help of the staff of the Naval Postgraduate School, specifically Prof. Alan Washburn and Captain Klaus Wirths of the Federal Republic of Germany, my knowledge of modeling the procurement process would certainly be more feeble than it is.

I cannot thank my mentors, Major Joseph Litko and Dr. James Chrissis, enough. They provided needed encouragement, direction, wisdom, and kept a dumb fighter pilot on the right research road. I deeply appreciated the assistance Prof. Gerald Brown, of the Naval Postgraduate School, offered to a student from another university. He freely shared his knowledge, his experience, and most importantly, his time.

Above all I am indebted to my three children; Michelle, Kathryn, and Daniel who raised my spirits nearly every day by being happy to see me no matter what hour of the

day or night; and to my wife, Jill, who endured a great deal to let me be a student once again (and for the absolute last time!)

Jack A. Jackson Jr.

Table of Contents

List of Figures

List of Tables

Abstract

The Theater Attack Model (TAM) is a large-scale linear program (LP) used to aid senior decisionmakers in making the tough budget and procurement decisions for the United States Air Force. TAM, as currently configured, can generate matrices with 9 million variables. The CPU-time to run this model is enormous. Advanced LP techniques are examined to reduce TAM's CPU-time.

A taxonomy of advanced LP techniques results from a review of major characteristics of these techniques. Promising opportunities to reduce TAM's solution time are tested and compared to a standard simplex benchmark. Karmarkar's algorithm on the KORBX (AT&T) system is tested and compared with simplex-based techniques through an absolute comparison of solution time and a relative comparison based on machine capabilities.

Recommendations for reducing TAM's CPU-time are outlined with the hope of saving the government money in computer time.

A TAXONOMY OF ADVANCED LINEAR PROGRAMMING TECHNIQUES AND THE THEATER ATTACK MODEL

I. INTRODUCTION

1.1 General Background

Annual decisions concerning procurement of aircraft, munitions, and spare parts are of enormous interest to senior military decisionmakers. Currently the United States Air Force (USAF) uses the Theater Attack Model (TAM), a large-scale linear program (LP), to evaluate theater level tactical air operations in support of procurement decisions. The Air Force Center for Studies and Analysis (AFCSA) maintains TAM and uses it to conduct trade-off analyses by examining the impact of the following factors: budget changes; aircraft and munitions effectiveness; target values; attrition rates; the costs of current and forecast aircraft, munitions, and spares; existing force structure of aircraft and munitions; weather; length of mission; and length of conflict. Currently, AFCSA is examining ways to expand TAM to include airbase operability and electronic countermeasures.

AFCSA runs TAM approximately 500 times each year. The model is being exported to analysis agencies and major commands interested in enhancing their understanding of the impact of budget constraints and the marginal values of different operational capabilities. The sheer size of TAM (generally 3500 rows by 250,000 columns) makes it one of the largest LPs run on a regular basis at the Pentagon. TAM, as currently configured, can generate matrices as large as 5000 rows by 8.75 million columns when considering classified global war situations. Current solution of the model requires 20–200 minutes of computer time (CPU-time), which equates to about 1–18 hours of clock time. In many instances, TAM must be rerun to obtain realistic operational answers; often, this is awkward due to time constraints and TAM's long CPU-time requirement.

1.2 A Short History Lesson

A review of the past will help put in perspective "how we got where we are." TAM is a relatively new model, but quantitative analysis has a long history within the weapons procurement process of the Department of Defense (DOD). The early days of linear programming started during and just after World War II with Project SCOOP, a military applications effort sponsored by the USAF and guided by George Dantzig. Within the procurement area, as early as the mid-1970s the British were using LP to optimize their weapons acquisition process (RAF CSNM). The DOD has been using linear and nonlinear techniques since the early seventies with the Navy's PHASER and NAVMOOR models and the USAF's family of models: SABRE MIX, FAST ATTACK, and HEAVY ATTACK [5] [23] [30] [47]. Several departments of the Air Staff, principally AF/XOXFM and AF/XOXJW, have ties with the Naval Postgraduate School, dating back to the early seventies in working these issues.

TAM grew out of an attempt in the mid-1980s to move away from strict weapons effectiveness studies on a micro scale to large, theater-level, parametric analysis [31]. Unaware of the previously mentioned work in this area, LTC Robert J. Might, the originator of TAM, created a model to measure the value of a particular weapon system or munition in the context of an entire conflict in a theater war. He identifies simple one-on-one weapons analysis as inadequate and suggests a senior decisionmaker needs insight to deeper issues:

> -How much of a weapon system's effectiveness can be covered by other systems? Which other systems?
>
> -How would attrition affect a current or proposed weapon system?
>
> -How effective is every weapon system and munition combination against every available target type?
>
> -How often will a particular weapon system and munition combination be turned in a war? How many of these sortie types can we realistically support?
>
> -How long will a weapon system and munition combination remain effective?

-If a new weapon system is not selected for production will its lost marginal effectiveness be replaced by another weapon system/munition combination?

-Would a new weapon system's entrance into the inventory remove all need for an older weapon system?

Might makes the point that these critical issues cannot be examined through strict weapons effectiveness studies where a weapon system is matched against a few others. There is a need to consider the entire context of the theater war, therefore TAM was conceived to consider a multi-period conflict, across multiple weather bands, over multiple sortie distances evaluating:

-effectiveness of each aircraft and munition combination against each target type,

-expected attrition of each aircraft against each target type,

-daily sortie rates for each weapon system,

-current inventories of aircraft, munitions, and spare parts,

-the numbers and values of enemy targets, day by day, including the effects of replacements,

-procurement costs of new aircraft, munitions, and spare parts,

-the value of spare parts to increase, decrease, or maintain sortie rates [31:55-63].

A current TAM formulation is included in Appendix 1.

Output from TAM allows parametric analysis of munitions and weapon systems so as to determine the marginal value of any single category and it's contribution to total operational effectiveness. Further, TAM is capable of solving the same problem with new or perceived budget adjustments to answer the questions about where the next dollars should be spent. Future add-ons to the model include the capability to consider airbase operability and the effectiveness of electronic countermeasures to aircraft survivability.

1.3 Research Objective

The research objective is to *identify advanced operations research techniques that will reduce TAM's CPU-time by 25%.* Advanced linear programming techniques will be surveyed for application to TAM. The advanced techniques that appear promising will be tested against TAM problems. While the research effort examines the TAM problem at AFCSA, the need to export any improvements to the wider USAF community will be kept in perspective when making recommendations. Any outstanding opportunities to reduce the CPU-time through computer hardware and software changes will be mentioned as appropriate, though specific hardware and software fixes were not direct research objectives. While the 25% criteria is an arbitrary goal, this research effort will endeavor to reduce the run time as much as possible.

1.4 Limits of the Research

The research effort considered a number of viable operations research (OR) techniques for reducing the run time of TAM. A taxonomy was formed with each technique assigned to a subset according to similar characteristics that categorize similar operation. The following is a complete list of advanced techniques considered as possibilities for application to large-scale LPs:

1. Special structures and basis factorization methods: initial basis crashing methods (sometimes called block pivoting), Dantzig-Wolfe decomposition, simple upper bounds (SUB), generalized upper bounding (GUB), generalized networks (GN) and pure networking (NET) techniques;

2. Interior methods: Karmarkar's algorithm, Khachian's ellipsoid method, Marsten's dual-affine interior point algorithm,

3. Restructuring methods: reformulation of the problem, goal programming, and a method from linear programming folklore.

The TAM model provided for study by AFCSA was the "small theater war" form of the model and uses unclassified input and output. The use of the smaller version of the model and selection of smaller TAM problems for testing were requirements dictated by

the lack of capability to handle extremely large bodies of classified data on the Air Force Institute of Technology (AFIT) computer systems. The largest of the TAM problems selected for testing was approximately 1/20th the normal TAM size. It is assumed that application of these techniques to smaller scale TAM problems will generate similar results on large-scale TAM problems. This appears to be a conservative assumption since larger TAM matrices would be very sparse (on the order of .1% dense) and several of these techniques expect to be relatively more effective on the sparser TAM problems. TAM comes in many forms depending on the specific theater under study, generally the modifications are to the data base supporting the model.

TAM, as currently configured, consists of 15,000 lines of FORTRAN 77 code that form a pre-processor, an interface to the MPS III linear programming system, and a post-processor. The pre-processor is a matrix generator that takes the data base and problem formulation and generates a file containing the problem matrix in MPS format. This file is passed to the linear programming system where the problem is solved and the solution is returned to the post-processor. The post-processor is essentially a report writer that compiles the results into a useable format for the analyst.

1.5 Computation Time

The research literature indicates several avenues to pursue when dealing with solution time and large-scale LPs. A number of researchers identify the number of rows contained within the matrix as the key to LP speed of solution. Dantzig reports the number of iterations is usually less than $(3m/2)$ and rarely rises to $(3m)$ where m represents the number of rows in the matrix [16:160]. Chvatal suggests for a fixed number of rows the number of iterations rises as a logarithm of the number of columns [14:45-51]. Bradley, Brown, and Graves suggest computations in the simplex method grow proportionally to m^3 and that certain structures allow this number to be efficiently reduced by a factor of nine through partitioning (basis factorization) and taking advantage of certain special structures that occur quite often in large-scale LPs [6:506-507].

Beale provides a detailed explanation of the calculations performed at each iteration of the primal simplex and product form of the inverse methods. He identifies several

computational advantages the inverse method has including:

-possibilities to exploit structure,

-compact representations of the basis inverse,

-flexibility concerning the transfer of data back and forth from computer real memory to computer storage,

-and flexibility about requirements to recompute the basis inverse.

Beale considers the use of a matrix generator key in dealing with large-scale LPs. He discusses in detail their positive contributions and concludes that very little CPU time is required to gain these benefits [2:66-96].

Garey and Johnson present a detailed discussion of the complexity issues that characterize the difficulty of problems and the efficiency of algorithms used to solve them. They provide a framework for classifying problems and algorithms. The following terms are defined in their work and are used throughout this paper:

Polynomial - (good) a class of problems that can be solved in polynomial time, therefore they are probably easier to solve than the following types of problems,

Exponential - (bad) a class of problems that cannot be proved to be solvable in polynomial time,

NP-complete - (bad) a class of problems that are considered hard (intractable), often these problems can be solved in exponential time [21].

There is much conjecture that the complexity of a problem is a good gauge of its intrinsic diffculty, and that an exponential algorithm (for instance) should not be necessary to solve a polynomial problem. This is the central thrust of criticisms of the simplex method (exponential) used for LP (polynomial). However, there is no definitive proof that the polynomial problems are, in fact, distinct from some classes of problems suspected of being harder (e.g.NP-complete problems). It is currently fashionable to establish the complexity class of a problem, or an algorithm, and leave the interpretation to the reader.

Perhaps someone will eventually prove that polynomial and NP-complete problems are distinct, perhaps not.

Given these comments, standards were developed to judge each method and any resulting benefits:

1. Total CPU-time will be the measure of effectiveness (MOE). Due to Beale's comments, TAM was unhooked from it's pre- and post-processors and matrix generation time is not considered in this research. The times reported compare the time for the LP solver to read in the problem file, solve the LP, and write the solution out to a file.

2. Though not an MOE, the number of iterations required for solution will be mentioned as appropriate. The amount of computational time required for an iteration in many advanced techniques varies, but often the number of iterations can provide insight for comparable techniques. Because certain techniques can increase the work associated with a single pivot (Karmarkar's method for example), it is possible to reduce the number of iterations yet increase total CPU time. Throughout the research, the $(3m/2)$ and $(\log n)$ guidelines were used as preliminary estimates to determine the feasibility of a method for reducing the CPU-time.

3. In the same manner, though also not an MOE, the tractibility of the solution technique *ala* Garey and Johnson will be used to consider an algorithms theoretical solution speed.

4. To scientifically measure increased performance a benchmark is required. An absolute comparison is a difficult, maybe impossible, task with different algorithms run on different computer hardware. Generally, a simplex benchmark was sought on the same piece of hardware; where this was unavailable the results are shared and absolute comparisons are made; relative comparisons are attempted, based on respective computer capability.

1.6 An Overview

The remainder of this paper presents the research and results. Chapter 2 is a literature review of the various OR techniques considered for this task. Each method will be described briefly before being applied to TAM. If a method was not applied to TAM

the reasons for it's non-application will be made clear. Chapter 3 covers the application of the selected techniques to TAM sample problems, outlines the actual tests run using selected techniques, reviews the results of the test runs, and makes observations from the results. Chapter 4 compares the best alternatives with respect to absolute solution time and relative solution times irrespective of the computer hardware used to solve the LP. Chapter 5 concludes the research with an examination of the best alternatives, their costs to implement, and closes with recommendations for AFCSA. Appendix A contains the AFCSA formulation for TAM and Appendix B is a suggested reformulation covered in Chapter 3.

II. OPTIONS

2.1 Introduction

A complete list of the advanced OR techniques considered for application to TAM will be examined in this chapter. It is the custom in academia to mention successes and not identify the bad detours research has covered. However, the lessons learned through disaster are often more valuable than successes because the "learner" has firsthand experience to remind him of the paths that bore no fruit. Often these side trips are excellent ideas awaiting the right opportunity. In this spirit, a taxonomy of possibilities is offered for attacking large-scale LPs so others can choose the right "tool".

A clear understanding of the simplex method is necessary to start. A complete description can be found in many basic texts on the subject (excellent tutorial choices include the texts by Chvatal [14] or Bazaraa and Jarvis [3], while Orchard-Hays [39] gives a detailed professional account). Throughout this paper small italicized letters represent vectors and capital italicized letters represent matrices. Terminology differs among authors and the choice of whether to maximize or minimize in canonical form is not standardized. TAM is formulated for "maximization" and the following discussion will proceed from that point of view.

In simple terms, the simplex method marches around the outside edges of a convex set, looking for improvement in the value of the objective function. The method terminates when it reaches a vertex (sometimes referred to as an extreme point, a point that cannot be formed from the convex combination of any other two points in the set) from which no further improvements can be made. Simplex problems can be expressed in matrix standard form as:

$$\begin{array}{rlll} \text{Maximize (or minimize)} & \sum_j c_j x_j & & \text{(the objective)} \\ \text{such that :} & \sum_j a_{ij} x_j & \leq b_i \ (i = 1,2,\ldots,m) & \text{(constraints)} \\ \text{with all} & x_j & \geq 0 \ (j = 1,2,\ldots,n) & \end{array} \tag{2.1}$$

The a_{ij} coefficients can be represented by the matrix A. The simplex method starts from the origin whether feasible or infeasible by introducing an auxillary variable for each constraint yielding

$$Ax + Is = b \tag{2.2}$$

where the identity matrix I is the same order as the number of rows in the original problem and is appended to the A matrix. The original variables x are decision variables and at the start are called non-basic. The new variables s are added to the vector x, are known as slack varibles, render the constraints strict equalities, and are called basic. The variables that are basic are said to form a basis for the convex region identified by problem 2.1. A basis is a linear combination of a set of vectors that can produce (span) all other points within the region. For convenience, the A matrix, the x vector, and the coefficients c in the objective function can be redefined to accomodate the new slack variables, and can be subdivided by association with the basic or non-basic portion of the x vector (i.e. $A_N, A_B, x_N, x_B, c_N, c_B$). Therefore, problem 2.1 is equivalent to:

$$\begin{aligned} & c_N x_N + c_B x_B \\ A_N x_N + A_B x_B &= b \\ x_N, x_B &\geq 0 \end{aligned} \tag{2.3}$$

Tradition identifies the basic portion of the A matrix as B, and, therefore B will be used in place of A_B . At any point in the simplex method the basic variables can be defined as a function of the non-basic variables:

$$x_B = B^{-1}b - B^{-1}A_N x_N \tag{2.4}$$

and the current solution designated by z can be defined as a function of the beginning problem and the non-basic variables:

$$z = c_B B^{-1}b - (c_N - c_B B^{-1}A_N)x_N \tag{2.5}$$

where $B^{-1}b$ represents the current value of the basic variables x_B.

After initialization, the simplex method executes a series of steps that together are termed a pivot (iteration, by some authors). The next step of the simplex method identifies a variable from the non-basic set to enter the basis. This is accomplished by checking the non-basic x_N for a positive $c_N - (c_B B^{-1} A_N)$ value. This step is known as pricing. Many rules exist in practice to identify the entering variable, but selection of an entering variable indicates an improving direction for the objective function. If no entering variable is identified, then stop, the current basis is optimal.

After the entering variable is identified, a search is conducted to identify a variable from x_B to leave the basic set. The controlling factor is the restriction that all values of the x vector must remain non-negative. A single variable among x_B will block the amount the objective function may improve. This variable is the leaving variable. At this point a pivot is initiated to exchange the entering and leaving variables. An iteration is completed when the required mathematics to update the current state of the problem are complete. The basic and non-basic portions of the problem can then be reordered to reflect the current status of the problem, and the process is started again to identify a non-basic variable to enter the basis.

A number of potential problems exist but only the problem of degeneracy will be discussed in this paper. Degeneracy occurs at a vertex where a variable is available to enter the basis and provides no increase in the objective function. This occurs because a number of constraints intersect at a particular vertex and the simplex method can have difficulty resolving which direction to move. Klee and Minty [29:159-175] constructed problems that show the simplex method is capable of cycling (failing to resolve which direction permits an improvement), but in practice this very seldom occurs. Wolfe discussed degeneracy and concludes that it is not a problem and that perturbations caused by numerical computation are generally sufficient to resolve these possibilities [48:205-211].

Basically the simplex method continues to search along the edges of a convex region in hyperspace. Once the solution to the LP becomes feasible it remains so. The intersection of two or more edges in hyperspace is known as en extreme point; it is at these points that

the simplex method checks for an improving direction. The simplex method terminates when a check for further improvement of the objective function suggests any change in the current basis will allow no further improvement in the objective function.

2.2 Special Structures and Basis Factorization

The growth of Linear Programming has been tied to the advances made in computers; without the machine the technique would not be practical. Improvements in LP have often been related to smarter computer applications increasing LP capability and reducing the CPU-time required for the simplex method. The standard simplex method updates the basic and non-basic partitions of the A matrix, current feasible solution, and computes the basis inverse at each iteration. Compact inverse methods were introduced to reduce the simplex method's computational load and take advantage of specific computer storage techniques. Among the methods capitalizing on special structure and/or a compact inverse are the revised simplex method (product form of the inverse), pricing and suboptimization, basis crashing, Dantzig-Wolfe decomposition, generalized networks (GN), pure generalized networks (NET), simple upper bounds (SUB), and generalized upper bounding (GUB).

The revised simplex method from the product form of the inverse (Dantzig and Orchard-Hays) demonstrates the growth of the simplex technique with the growth in computer technology. Its basic philosophy is to compute current prices and entering column coefficients only as needed. The product-form inverse methods provide a simple way to update B^{-1} by post-multiplying the current basis inverse by an eta matrix. The eta matrix E is a compressed identity matrix with a single column altered to record the actions taken in a single pivot. Post multiplication of the current basis inverse by E accomplishes a pivot of the simplex method.

The Revised Simplex method does not execute every step of the simplex method at each pivot. It prices the columns, selects the most promising, saves the change in basis through an eta matrix, and stores the eta matrices in an eta file along with the initial basis until a predetermined limit of eta matrices are reached. At this point, a reinversion occurs and the current basis, current feasible solution, and current basis inverse are updated

through knowing the initial problem statement and the series of eta matrices that reflect the history of pivots [17:64-67].

Beale identifies three benefits the revised simplex method uses to reduce the CPU-time required to solve an LP. Though the standard simplex method has fewer calculations per iteration, the revised method saves computational time on large-scale models when a large number of the elements of the matrix are zeros. Typically, the density of non-zero elements in large-scale models is on the order of 0.1%. The computational savings occur in conjunction with pricing techniques (multiple or partial, covered later in this section) by not executing all of the simplex operations at each iteration—principally not computing $c_N - (c_B B^{-1} A_N)$ for all x_N. Secondly, the revised simplex method reduces the substantial computer memory required by packing B^{-1} into a smaller array. This allows a larger matrix to be worked on in real memory and when paging (computer retrieval of information from disk storage to the working memory) occurs, a smaller array is passed to and from main memory. The third benefit of the revised simplex is that it allows control over the representation of B^{-1} and how it will be manipulated in the computer. This control further reduces paging back and forth from real memory to disk storage within the computer.

A final benefit related to accuracy is the revised simplex method allows mathematical calculations to be accomplished which reliably detect round- off error and permits reinversion so that numerical inaccuracies are not accumulated and carried forward. Current large-scale computer programs use some form of the revised simplex method and refer to the primal simplex method which includes both a phase 1 (search for feasibility) and a phase 2 (search for optimality) [2:66-96] [3:194-195].

Orchard-Hays, in his classic work on LP computational techniques, describes multiple and partial pricing as techniques to capitalize on revised simplex column pricing by selecting a subset of promising vectors for inclusion in the basis. Multiple pricing selects a best subset of the $c_N - (c_B B^{-1} A_N)$ vector, performs simplex operations on them one at a time, and suboptimizes the current problem. Partial pricing selects several columns from $c_N - (c_B B^{-1} A_N)$ and enters them into the basis all at one time. Early in the solution of a large system most of these more promising columns represent movements in an improving

direction. Therefore, forcing their entry into the basis all at once can lead to computational savings and reduce the infeasibilities of the problem. These techniques allow for a return to the primal simplex method when preconditions are met which indicate pricing (partial or multiple) will no longer be effective. Note that the effort expended in pricing is generally rewarded in fewer iterations, however there is some trade-off in terms of total CPU-time between greedy pricing with many iterations and scrupulous global pricing with few iterations [39:109-114]. Most modern LP solvers maintain an automatic capability to introduce partial and multiple pricing.

McBride generalizes many of the following techniques under the banner of "factorization". The main points of these methods include implicitly carrying portions of the inverse basis that can be generated from a reduced explicit basis that is used for all operations. He suggests the following advantages of these methods:

> -good initial bases (starting points),
>
> -reduced computational workload as a result of operating with a reduced basis,
>
> -further CPU-time savings through reduced core storage requirements and less paging to and from disk storage [33].

The following methods are based on finding specific structures within the TAM matrix, taking advantage of these special structure in the primal simplex method, and specific techniques for handling the basis inverse.

2.2.1 Crashing the Basis (Gaining a Good Initial Basis). Orchard- Hays identifies crashing the basis (block pivoting) as an attempt to maximize the early steps toward the optimum without regard to the infeasibilities that may be introduced. Cooper outlines the concept as based on selecting a set of legitimate contenders from the non-basic set and using these to replace current basic (slack) variables. The objective is to select and replace as many slack variables as can be rapidly handled. If, at the outset, it can be determined that a portion of the matrix is always basic or a subset of variables is generally basic, then crashing these subsets into the basis might reduce the total number of iterations required as

well as the mathematical overhead at the start of the normal simplex application [39:136-139] [15:144-151].

Nazareth points out that the suggested starting basis needs to be well conditioned (non-singular) and as close to feasible as possible [36:163]. Wolfe and Cutler's research identifies a specific formula for selecting a good initial basis. They conclude that the initial basis should be selected from singletons first (variables which appear in a row as the only non-zero entry), then from slack variables with a positive value on the right hand side of the linear constraint, and finally from slacks with a zero right hand side. Their results suggest that success with crashing is a function of each specific LP — bad crashes are terrible, good crashes are great [48:177-200].

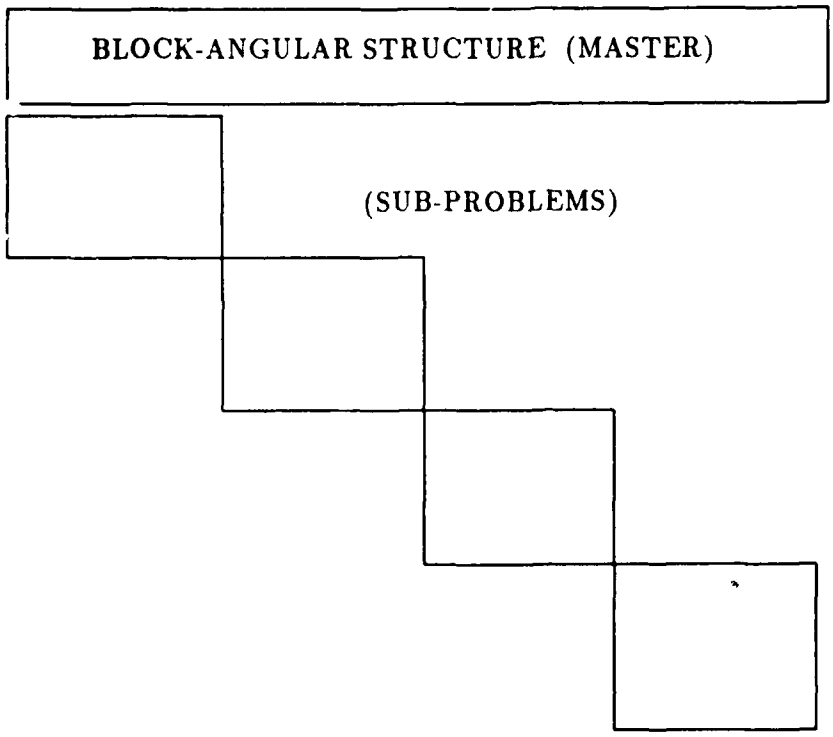
2.2.2 Dantzig-Wolfe Decomposition. With the growth of LP as an accepted analytical technique in the 1950s and 1960s, many large linear programming problems arose which could not be accommodated by computers of that day. Dantzig and Wolfe devised a computational technique to take advantage of special structures that often occur in practice. This technique decomposes the original problem into sub- problems represented by the smaller matrices along the diagonal.

In practice, many large-scale LPs are formulated by linking a series of smaller subsystems together with a set of master constraints that bind the small problems together into a larger whole. Decomposition separates the common constraints from the portion of the matrix containing the special structure. The special structure is characterized by no subsystem having any relation to any other subsystem apart from the common constraints.

The constraints common to all columns remain to form a master problem. The master and the sub-problems are then linked through rewriting the sub-problems as weighted, convex combinations of the extreme points of the sub-problem feasible regions. A modified problem is then formed based on these weighted proposals to the master problem; this new problem is called the restricted master problem. The restricted master problem is formed with fewer rows, but has a columns section that can increase dramatically.

Transfer of information between the sub-problems and the master problem occurs at each iteration through to solution. The sub-problems form new proposals for allocating the

Figure 2.1. Block-Angular Matrix Structure



variables comprising the minor iterations of the method, and submit this information to the master problem. The major iterations occur as the master problem considers the proposals from the sub-problems and reprices the variables. As information is passed between the master and sub-problems they tend to converge until the optimal value of the objective function is found.

Dantzig identifies a popular economic interpretation arising from decomposition. Viewed as subsystems (sub-problems) striving for scarce economic resources provided by headquarters (master problem), the initial plan of the subsystems is fed to the master planner who weighs the proposals. He assigns new prices for the scarce resources forcing the subsystems to revise their initial plans. The subsystems evaluate the new prices, create new proposals, and input these to the master planner who continues the cycle. When modification of the inputs or prices is no longer required, the system has reached optimality and, in turn, the original LP is solved [16] [19:101-111].

Though the decomposition principle has interesting theoretical value, and still to this day is used when computer memory is over-taxed by an enormous problem, it does not generally lead to quicker solutions. Beale suggests that as long as the common row section is maintained between 50 and 100, this method could lead to quicker solution times [2:169-170]. The literature suggests this technique should be viewed as a complement to the simplex method and it is best suited to situations where computer memory and size limitations are pushed to the limit. Therefore, Dantzig-Wolfe decomposition was not applied to the TAM problem. It remains the best method for dealing with oversized problems that tax the limits of a computer's memory allocation, especially when reformulation is not appropriate.

2.2.3 Embedded Pure (NET) and Generalized (GN) Networks. McBride suggests factorization of the basis (isolating portions of the A matrix with known special structures) as a comprehensive framework to view computational methods of large-scale LPs. He proves factorization is useful for reducing the computational workloads of large-scale LPs [33]. Factorization techniques include :

> *Simple bounds* - rows with a single non-zero coefficient.
>
> *Generalized Upper Bounds* - a set of rows for which each column contains at most a single non-zero coefficient.
>
> *Generalized Network Rows (GN)* - a set of rows for which each column contains at most two non-zero coefficients. If the nonzero coefficients are restricted to be a (+1), or a (-1), or both, then the structure is known as an *embedded pure network (NET).* This section considers the latter technique.

Brown and Wright, in two articles on identification of embedded networks and extraction of embedded networks, identify the structures mentioned above as possibilities for exploitation that lead to quicker solution time in large scale LPs. The identification of maximum subsets of these is itself an NP- complete problem. Therefore, Brown and Wright suggest heuristic search methods that provide maximum or near-maximum GUB, GN and NET subsets. These methods are modifications of the heuristic search mentioned

in the GUB section and identify these special structures in very short periods of time. The articles conclude with a series of samples tested for GN and NET structures and positive test results [10:41-46].

Olson gives a complete description of these methods and reports the results of GN and NET testing on TAM sample problems. His dissertation provides a good general code for discovering a GN or NET structure, exploits this structure during solution, and reduces the time required to solve problems with these specific structures [38].

2.2.4 Simplex Method for Bounded Variables. In LP variables are often bounded having constraints of the form:

$$l_j \leq x_j \leq u_j \qquad (j = 1, 2, \ldots, n) \tag{2.6}$$

Slack vectors can be introduced for the upper and lower bounds, but this increases the size of the problem to $(m + 2n)$ constraints and $(3n)$ variables. If $l_j = 0$, then we have the standard non-negativity constraints. Since $x_j \leq u_j$ is a series of normal LP constraints, the simplex method for bounded variables by Dantzig was devised to efficiently deal with this specific problem.

With upper bounded constraints, it is determined beforehand that any set of variables x_j that has an upper bound will be non-basic whether at 0 or at its upper limit. To ensure this a new set of variables x'_j is introduced and defined as $x'_j = (u_j - x_j)$. This forces the new variable to equal zero whenever the original variable is at its upper bound and vice-versa. Three cases arise during the implementation of SUB worth mentioning in relation to constrained variables as they become targets for entering the basis:

> *Case I.* If x'_j reaches it's upper bound, then a substitution is made replacing x'_j with $(u_j - x_j)$. No variables enter or leave the basis.
>
> *Case II.* If x'_j increases to a point were it causes a basic variable to leave the basis, then x'_j enters the basis in the place of the variable that was forced to it's lower bound. The variable replaced becomes non-basic, hence a normal pivot.

Case III. If x'_j increases to a point where it forces a basic variable to it's upper bound, then x'_j enters the basis and the variable forced to it's upper bound becomes non-basic at it's upper bound and we perform the Case I substitution on the leaving variable [16].

As Case I requires no change in the basis and Case III accomplishes a basis change with reduced computations, this method can produce situations where the number of iterations (basis changes) remains nearly constant with a decrease in the number of calculations performed and a subsequent decrease in CPU-time.

2.2.5 Generalized Upper Bounding. The GUB special structure of Dantzig and Van Slyke requires that a set of rows have at most a single non-zero element in each column [18:213-226]. The key to GUB is recognizing that at least one of the columns with a nonzero-element in a GUB row must be basic, this column is the key column. Generally, the non-zero elements are scaled to unity aiding the computational efficiency of the method. Chvatal estimates that if 30% of the rows have a GUB structure then this method will be faster than the revised simplex method. When the number of rows exhibiting a GUB structure reaches 80% the GUB method requires one-tenth the time of the simplex method [14:416].

Chvatal description of GUB considers a problem where there are m rows and n columns. If the GUB structure has p rows then the GUB method maintains a working basis B of at most dimension $m' = m - p$. The bookkeeping for the basis B' will be substantially reduced when the GUB structure p is large relative to the total number of rows m. The GUB structure permits partitioning of the basis in a special manner:

Note that the submatrix A contains m' rows and p columns and, along with the identity matrix I, form what are called the key columns. The submatrix B called the kernel of the GUB method is m' by m'. The bottom half of the matrix are the GUB rows rearranged to form the submatrix I, an identity matrix of order p. The submatrix C then contains the remaining columns of the GUB row set. It can be shown that a reduced working basis of $B' = B - AC$. The advantages of the GUB method are a novel method of updating the reduced basis, reduced storage requirements by operating with a reduced

B' =

B	A
C	I

basis, and the reductions in computations required at the expense of extra bookkeeping.

Given this structure for the basis, it can be shown that at each iteration of the GUB method one of three possible cases can exist:

Case 1. If the leaving column is not a key column; then replace the leaving column with the entering column and multiply the current working basis by a suitable eta matrix.

Case 2. If the leaving column is a key column and some column of the working basis is equivalent to the leaving column in the GUB row section, then insert the entering column in the place of the leaving column, multiply the current working basis by a suitable eta matrix, and multiply the current basis by a p-order identity matrix with a row updated.

Case 3. If the leaving column is a key column and no column of the working basis is equivalent to the leaving column, then replace the leaving column with the entering column.

After a number of iterations the basis inverse is simply a partitioned matrix of the original working basis inverse multiplied by a series of eta matrices

$$B^{-1} = (B - A \cdot C)E_1E_2E_3 \ldots \tag{2.7}$$

Solution times are shorter because the reduced computational workload afforded by a reduced working basis compensates for the extra bookkeeping effort required to remember which columns have switched in and out of the key column section [14:415-424].

2.3 Interior Methods

Two recent developments in the field of linear programming, Khachian's ellipsoid method and Karmarkar's algorithm, have stirred interest with the promise of solving large-scale LPs in polynomial time. These interior methods have spawned an entire generation of articles and research in their application. A third technique considered in this section, Marsten's dual- affine interior point method, is an expansion of these ideas. While the verdicts on Karmarkar's algorithm and Marsten's method are still out, experience has shown that the simplex method is superior in almost all situations to Khacian's ellipsoidal technique.

Many of the specifics of Karmarkar's and Marsten's methods and the machines the algorithms run on cannot be reported as they are maintained as proprietary secrets. Research does indicate that both of these methods run on parallel vector processing machines. Most computers function by carrying out a single computation at a single instant in time, this process is known as sequential processing. Vector processors belong to the growing class of parallel processing machines that conduct simultaneous or concurrent operations. Vector processors operate on entire vectors instead of calculating each computation one element at a time. When this is done with parallel co-processors, this reduces computation time and the algorithms can be adjusted to take specific advantage of the hardware's numeric characteristics [43:357-382].

2.3.1 Khachian's Ellipsoid Method. In 1979 at the Mathematical Programming Symposium in Montreal a translation of Khachian's method was first presented. He caused quite a stir by proposing a method for solving convex programming problems using a series of ellipsoids to iteratively converge on the optimum in polynomial time [28]. As LP is a special case of convex programming, there were immediate predictions that this technique would make the simplex method obsolete, since the best known bound for the simplex oper-

ation is exponential. Although simplex- based methods may in theory require exponential time, in practice they are solved much quicker than that.

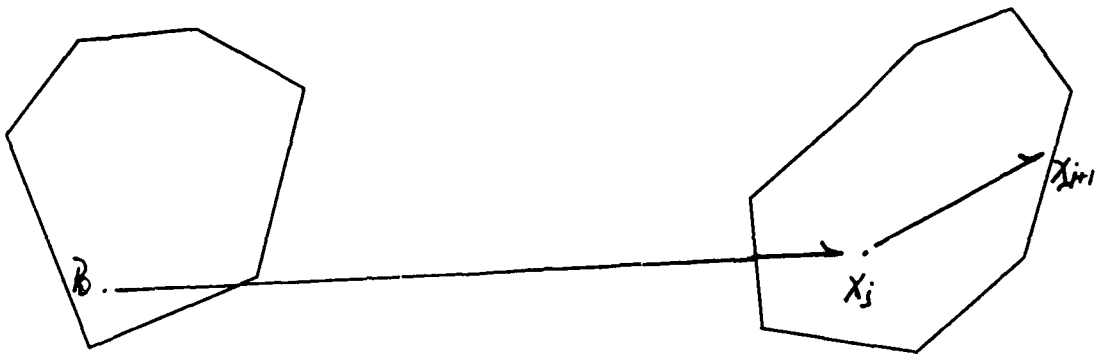
Predictions of the death of simplex were premature as experience shows the simplex method to be more robust and almost always significantly faster than Khachian's method. Khachian's method did however prove to be a significant discovery in the sense that the LP problem is established as polynomially complex, the easiest class of problems, in terms of pure mathematics. Since the simplex method is exponential, it is in theory, at least, an inefficient method for LP and a fertile field from which researchers have launched new projects in applied mathematics [14:443-454].

Periodical literature is peppered with numerous studies spinning off from this method. With further research and revision it may yet provide a breakthrough for solving convex (and therefore LP) problems significantly quicker. Because the weight of current research suggests this method will not solve an LP problem faster, and because no excellent computer codes applying this technique are commercially available, this method was not attempted.

2.3.2 Karmarkar's Algorithm. Karmarkar's algorithm is a new theoretical innovation that, starting from an interior feasible point, proceeds through the interior of the convex region to successive points converging to the optimum. This method has been proven to have polynomial time complexity for LP. From the inner point a projective transformation maps the convex region into a similar region in another convex region with dimensionality increased by one. At each step the method searches for the most promising direction of improvement; a line search is conducted to detect the maximum step size in the direction of the gradient; and some portion of this step size is taken as the real improvement in the current solution. The method stops after successive steps return sufficiently small changes in the objective function value. The reason this method has drawn so much attention is that the repeated application of projective transformations optimized over the inner sphere creates a series of points that converge to the optimum in polynomial time [26].

The main steps of the method (from Strang) are:

Figure 2.2. Projective Transformation



PROJECTIVE TRANSFORMATION

1. Identify a point interior to the convex region. An artificial variable is added to the matrix so that a point P_0 is found, where all original variables (x_j , $j = 1, \ldots n$), are equal to 1 and P_0 is interior to the convex region.

2. A check is made to determine if the current objective function value is within some distance δ of the optimum (the KORBX system determines δ by a check of the difference between the primal and dual optimal solutions). If yes, then stop; else continue.

3. A diagonal matrix D is constructed such that

$$D - 1^T x_j = (1, 1, \ldots, 1) = E \tag{2.8}$$

4. Using D, compute the projection

$$P = Dc - DA^T y \tag{2.9}$$

5. Find a step size s such that $E - s(P) = 0$.

6.. Reduce the step size s by some factor so that the step remains within the feasible region rather than on the boundary (say 0.90).

7. Compute the new initial vector x_{j+1} by

$$x_{j+1} = x_j - sD(P) \tag{2.10}$$

The new vector x_{j+1} represents the new point which remains within the interior of the region. At this point return to step 2 and continue.

The keys to Karmarkar's method are the rescaling and the transformation projection. The transformation projection produces most of the computational work in this method. The compelling geometric argument for this method is that the rescaling presents a better view of the constraints for better global convergence [44:405-410].

2.3.3 Dual-Affine Interior Point Algorithm. Dr. Roy E. Marsten and an international group of researchers conducted research into a dual-affine method based on the paper of Adler, Karmarkar, Resende, and Veiga [1]. Due to the proprietary nature of the product, no citations or discussion of the general mathematics involved is available.

A few general comments are:

-The method is an interior point algorithm which handles simple upper bounds, ranges, and free variables.

-During execution of the method variables headed for zero are fixed there, and rows with slacks bounded above zero are removed.

-Initially the Big-M method of Bazaraa and Jarvis is used to gain feasibility.

-At optimality the simplex method is used to gain an optimal basis. This portion of the algorithm appears to take the largest part of the computation time.

This method was not tested on TAM problems due to logistical problems with sending and receiving computer tapes through intermediaries at MAC headquaters.

2.4 Restructuring TAM's Matrix

The methods of this section are related indirectly in that each requires structural changes to TAM. These methods would necessitate changes to the underlying FORTRAN code of the matrix generator. In some cases the changes are minor but others could encompass a wholesale revision. This would not necessarily be a bad thing since the

10,000 lines of code are poorly documented, replete with undefined variables, and basically unreadable to even the trained analyst. A complete rewrite could accomodate future expansion of the model and bring the following techniques to bear on the problem of speed. An alternative is the use of the new programming languages that would do away with MPS format (eg. GAMS, AMPLE, EML etc...).

2.4.1 Reformulation. This addresses the value of remodeling TAM and the theater war process, with the aim of reducing CPU-time. Essentially, this usually means reducing the number of rows and columns, thereby reducing the number of iterations (remember the $(3m/2)$ and $(\log\ n)$ relationships). Tomlin identifies possible side benefits from reformulation in a study he conducted on scaling. These include rescaling the matrix coefficients to prevent numerical problems during the solution, reducing the possibilities of singular matrices, and perhaps reducing the solution time. He concluded the major benefit was increased numerical accuracy during the LP, particularly in poorly formulated models. His results dealt with models created by professionals and he was unable to identify any specific trends toward time savings [45:147-165]. TAM appears to be well thought out; therefore, proposed changes would include slight reductions in the structure, removals of redundant sections, and checking for possibilities to rescale the A matrix coefficients.

Reformulation may also be viewed as an opportunity to enhance certain characteristics of the LP. For instance, given a code that took advantage of GUB structures it would be advisable to rewrite the model to enhance this portion of the matrix structure. Study in this area has made apparent three rules to keep in mind before formulating a large-scale LP:

1. Know your computer hardware, it's limitations and capabilities.

2. Know your software code, it's limitations and capabilities.

3. Formulate and use the matrix generator to take maximum advantage of these pluses and minuses.

2.4.2 Goal Programming. This method is used in situations when competing objectives or goals exist, and there is a need to resolve ties or rankings among the separate

objectives. Currently, AFCSA uses this method with TAM to analyze goals other than maximizing the Target Value Destroyed (TVD). Goal programming, historically, has not reduced the solution time since it adds constraints to a problem as it introduces a new objective function and is used iteratively to resolve rankings among objectives. A notable exception is the special elastic simplex algorithm by Brown and Graves [8].

Winston has an excellent chapter on the application of goal programming and it's relation to multi-attribute theory. A hierarchy among the goals is established through the use of penalties and/or ranks. The method assigns penalties to the objective function for deviations from the goal, or ranks the goals and iteratively runs the simplex method introducing new goals at the start of each new simplex run, much like Phase 1 and Phase 2, continued for more phases. After each run of the simplex, the goals that are reached are inserted into the new LP as equality constraints that require these values as aspirational levels. Then the next hierarchial objective function is added and the simplex method is reapplied. Future simplex runs do not change the values of higher priority goals already maximized. This guarantees that each successively less important goal will not move the simplex solution away from previous optimal results for the more important goals. The simplex terminates when improvement in the objective function can only come as a result of lowering the value of a higher priority goal, or when the list of new goals to add to the model is exhausted [46:609-621].

Goal Programming can be used to restrict TAM for short-suspense analysis jobs where appropriate models do not exist and there is no time to construct a new model. Currently, AFCSA uses this technique by fixing the value of Target Value Destroyed (TVD) and adding a new objective function. As this method adds constraints to the matrix any reduction in time is through sheer luck, not wisdom. This technique can be expected to increase TAM's solution time, and was not tested.

2.4.3 The Folklore Approach. Due to natural curiosity and a belief in "old wives tales" a method by Charnes *et al* [12] was tested. This method is mentioned in the conclusions as an example of the ability to forget good ideas that work, while searching for new improved ways that do not work. The entire knowledge found on this subject is contained

within a single paragraph of the report:

> "Finally, in the earliest days of LP computation oil company computer groups observed that they got excessive times in general LP calculations whenever there were zeros in the right hand side vector. Computability solution: Introduce a new variable and equation constraint, $x_n + 1 = 1$ and add multiples of it to the constraints with zeros to knock them out. Result: great improvement in computation time. Although this device is part of the folklore, many newcomers to computation are still unaware of it." (Charnes)

Beale mentions a technique similar to this one in a discussion on degeneracy. The difficulty was encountered with large-scale LPs with numerous zeros in the right hand side vector. He suggests changing the constant terms slightly and restoring the values of the constants after solution [2:18]. This technique, known as perturbation, is often sighted by other authors as a method to overcome degeneracy [48].

Wolfe, in his study of pivot selection rules, states that degeneracy has little affect on modern codes, since when it is detected, the algorithms elect to pivot based on the size of gain. The concept of selecting the pivot based on the greatest gain does little to change the total number of pivots but helps to maintain numerical accuracies, along with other frequent reinversions of the basis and controls of round-off error [48:205-211]. Harris developed new criteria for selecting the entering basic variable based on dynamic weighting factors to estimate the maximum gain. Her results on large-scale LPs show an improvement in numerical accuracy and improvement in the total number of iterations. Her method does cost in terms of extra calculation per iteration but seems to introduce more of the final basis earlier and therefore results in overall savings of 50% [24:30-57].

2.5 Summary

The literature has suggested a number excellent avenues to pursue towards a reduction in the computational time taken to run TAM. A review of the best possibilities to reduce the total time taken to run and solve TAM suggest that the following techniques should be tested on TAM sample problems:

1. Basis Crashing
2. Simple Upper Bounds

3. Generalized Upper Bounds

4. Pure Network (NET) techniques

5. Karmarkar's Algorithm

6. Reformulation

7. The Folklore Technique

The next chapter considers the specific application of these techniques to TAM.

III. APPLICATIONS AND RESULTS

3.1 Introduction

This chapter outlines the experiments performed on TAM and the test results. Background material will be covered on the specifics of the TAM model, the sample problems selected as test cases, and the numerous computer hardware and software details important to understanding the tests and their results.

3.1.1 The TAM Model TAM's large-scale matrix is "very wide and not very deep". This is because the rate at which the number of variables grows is far more rapid than the rate at which the number of constraints do. The main decision variables X_{ijklmn} represent sorties where aircraft i, with munition j, is assigned to target k, in time band l, in distance band m, under weather band n. Since there are 20 aircraft, 58 munitions, 86 targets, 4 time bands, 4 distance bands, and 6 weather bands, there could be 8.75 million different variables. The other decision variables (NAC, NMN, and $NSPR$) represent the number of new aircraft, munitions, and spares purchased respectively. A short version of the formulation is listed here with the expanded version in Appendix A.

Objective function:

$$\text{Maximize} \quad TVD = \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n X_{ijklmn} \cdot YLD_{ijklmn} \cdot TGTVAL_{klm} - BAF \tag{3.0}$$

Subject to:

$$\sum_j \sum_k \sum_l \sum_n \frac{X_{ijklmn}}{TS_{ijklmn}} = EAC_{im} + NAC_{im} \qquad \text{for each } i, m \tag{3.1}$$

$$EAC_{i1} \leq STARTAC_i \cdot FILL_{i1} \quad \text{for each } i \tag{3.2}$$

$$EAC_{im} \leq STARTAC_i + \sum_m FILL_{im} + \sum_{m-1} NAC_{im} - \sum_{m-1} ATTRIT_{i,m} \qquad \text{for each } i, m, n \tag{3.3}$$

$$\sum_j \sum_k \sum_l \frac{X_{ijklmn}}{TS_{ijklmn}} \leq PRCNT_n \cdot \left[STARTAC_i + \sum_m FILL_{im} + \sum_m NAC_{im} - \sum_{m-1} ATTRIT_{im} \right] \text{ for each } i, m, n \quad (3.4)$$

$$\sum_i \sum_k \sum_l \sum_m \sum_n X_{ijklmn} \cdot LOAD_{ijl} = EMN_j + NMN_j \quad \text{for each } j \quad (3.5)$$

$$EMN_j \leq STARTMN_j \quad \text{each } j \quad (3.6)$$

$$KILL_{klm} = \sum_i \sum_j \sum_n X_{ijklmn} \cdot YLD_{ijklmn} \quad \text{for each } k, l, m \quad (3.7)$$

$$KILL_{klm} \leq TGT_{klm} \quad \text{for each } k, l, m \quad (3.8)$$

$$TGT_{kl1} = STARTGT_{kl1} \quad \text{for each } k, l \quad (3.9)$$

$$TGT_{klm} = TGT_{kl(m-1)} - KILL_{kl(m-1)} + REGEN_{klm} \quad \text{for each } k, l, m > 1 \quad (3.10)$$

$$\sum_i \sum_m (NAC_{im} \cdot ACCOST_i) + \sum_j (NMN_j \cdot MNCOST_j) + \sum_i \sum_m \sum_p (NSPR_{imp} \cdot SPRCOST_{ip}) \leq BUDGET \quad (3.11)$$

$$\sum_m ATTRIT_{im} \leq MAXATT_i \cdot \left[STARTAC_i + \sum_m FILL_{im} + \sum_m NAC_{im} \right] \quad \text{for each } i \quad (3.12)$$

$$\sum_i [EAC_i + NAC_{im}] \geq MINSORT_m \cdot \left[\sum_i STARTAC_i + \sum_i \sum_m FILL_{im} + \sum_i \sum_m NAC_{im} - \sum_i \sum_{m-1} ATTRIT_{im} \right] \text{ for each } m \quad (3.13)$$

$$\sum_i KILL_{klm} \geq MKILL_{km} \qquad \text{for each } k, m \quad (3.14)$$

$$NMN_j \leq UMN_j \qquad \text{for each } j \quad (3.15)$$

$$NMN_j \geq LMN_j \qquad \text{for each } j \quad (3.16)$$

$$\sum_m NAC_{im} \leq UAC_i \qquad \text{for each } i \quad (3.17)$$

$$\sum_m NAC_{im} \geq LAC_i \qquad \text{for each } i \quad (3.18)$$

$$BAF = e_a \cdot \sum_i \sum_m NAC_{im} + e_m \cdot \sum_j NMN_j + e_s \cdot \sum_i \sum_m \sum_p NSPR_{imp} \quad (3.19)$$

$$ATTRIT_{im} = \sum_j \sum_k \sum_l \sum_n X_{ijklmn} \cdot a_{ijklmn} \qquad \text{for each } i, m \quad (3.20)$$

$$NOTUSED \quad (3.21)$$

$$REGEN_{klm} = \sum_{z=1}^{m-1} b_{kzm} \cdot KILL_{klz} \qquad \text{for each } k, l, m > 1 \quad (3.22)$$

$$\sum_j \sum_k \sum_l \sum_n X_{ijklmn} = NSORT_{im} \qquad \text{for each } i, m \quad (3.23)$$

$$\sum_j \sum_k \sum_l X_{ijklmn} = PRCNT_n \cdot NSORT_{im} \qquad \text{for each } i, m, n \tag{3.24}$$

$$TVD = \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n X_{ijklmn} \cdot YLD_{ijklmn} \cdot TGTVAL_{klm} \tag{3.25}$$

$$\sum_l \sum_m KILL_{klm} \geq MKILL_{km} \qquad \text{for each } k, m \tag{3.26}$$

$$\sum_m ATTRIT_{im} \leq MAXATT_i \qquad \text{for each } i \tag{3.27}$$

$$\sum_l \sum_n X_{ijklmn} = AGSORT_{im} \qquad \text{for each } i, m, j = 58, k = 86 \tag{3.28}$$

$$\sum_l \sum_n X_{ijklmn} = AASORT_{im} \qquad \text{for each } i, m, j = 58, k = 86 \tag{3.29}$$

$$\begin{aligned} \sum_j \sum_k \sum_l \sum_n X_{ijklmn} \;\leq\; & \sum_m (INSPR_{imp} + NSPR_{imp}) \\ & - \sum_j \sum_k \sum_l \sum_{m-1} \sum_n X_{ijklmn} \quad \text{for each } i, m, p \end{aligned} \tag{3.30}$$

$$\sum_m NSPR_{imp} \leq UPPSPR_{ip} \qquad \text{for each } i, p \tag{3.31}$$

$$\sum_m NSPR_{imp} \geq LOWSPR_{ip} \qquad \text{for each } i, p \tag{3.32}$$

From the condensed formulation observe that:

-constraints 3.1, 3.2, 3.3, 3.13, 3.23, 3.24, 3.27, 3.28, and 3.29 represent aircraft and sortie availability restrictions,

-constraints 3.5 and 3.6 represent weapons availability restrictions,

-constraints 3.7, 3.8, 3.9, 3.10, 3.14, 3.22 and 3.26 represent target availability restrictions,

-constraint 3.11 is a budgetary restriction,

-constraints 3.12, 3.20 and 3.27 reflect attrition considerations,

-constraints 3.15, 3.16, 3.17, 3.18, 3.31 and 3.32 represent upper and lower bounds on the decision variables,

-constraint 3.19 calculates the budget awareness factors, constraint 3.25 is used in goal programming situations, and constraint 3.30 represents sorties restrictions due to spare parts.

3.1.2 TAM Sample Test Problems TAM comes in several scenarios with different databases for major combat theaters. Due to the magnitude of the standard TAM matrix, representative smaller models were selected reflecting the structure of the normal problems. TAM contains numerous optional constraint blocks that are "turned on" depending on the specific analysis project. Table 3.1 gives a view of the normal size problems and presents several normal scenarios.

Table 3.1. Representative TAM Problems

NAME	VARIABLES	CONSTRAINTS	
TAM ORIGINAL	8.5 million	5287	Global War
NATO SENSICAL	1.8 million	3676	Theater War
WEAPONEER	180,000	3676	
OPTIONAL CONSTRAINTS	180,000	3576	(50%)
NO EXTRA CONSTRAINTS	180,000	3037	
WITHOUT CONSTRAINT #7	180,000	2077	

Major Nick Reybrock of AFCSA provided expert assistance in selecting several sample problems for research. The problems reflect standard TAM matrices on a smaller scale. Table 3.2 identifies each problem and outlines pertinent data about the problem.

Table 3.2. TAM Test Problems

PROBLEM NAME	ROWS	COLUMNS	NON-ZERO ELEMENTS	DENSITY
PROBLEM.1	91	116	1,011	9.58%
PROBLEM.2	180	664	6,774	5.67%
PROBLEM.3	269	2,076	22,035	3.95%
PROBLEM.4	211	694	6,788	4.63%
PROBLEM.5	438	9,217	98,232	2.43%
PROBLEM.8	420	4,844	51,592	2.54%
PROBLEM.12	629	15,906	172,310	1.72%

The problems chosen for analysis in this report cover several standard options available to the modeler at AFCSA. Problems 1, 2, 3, and 5 are models where the problem number indicates the number of aircraft, munitions, and targets within the model. These are examples of smaller TAM versions used to analyze effectiveness of certain weapon systems compared to like systems. This application mirrors standard weapons effectiveness studies, matching a limited number of systems against each other. Problems 4, 8, and 12 are examples of models that conform to the standard large- scale modeling done with TAM. This application is TAM's major reason for existence. In all of these cases, the ratio of aircraft to munitions to targets is 1:2:4, with the problem number identifying the total number of targets [41].

3.1.3 AFCSA Computer Hardware and Software Considerations AFCSA uses MPS III (Ketron Inc.), a commercial linear programming package to solve TAM on an IBM 3084 mainframe. Smaller versions of TAM are solved on a VAX 11/780 using the SCICONICS software package (Ketron Inc. is the USA distributor for a British firm, Scicon Limited). The agencies to receive an export version of TAM use the SCICONICS package on a VAX or equivalent machine. SCICONICS is a reliable primal simplex implementation that uses the Forrest-Tomlin updating technique for added speed [20:263-278].

MPS III, available for large IBM mainframes, contains several packages for LP and is the current industry standard for solution speed. It contains a basic primal simplex system

which uses a matrix inverse packing technique to take advantage of sparse structure in the basis inverse. MPS III has capability to add-on a GUB module for use with large LPs that contain the appropriate sparse structure. AFCSA, at this time, has not purchased the GUB module. The current purchase price of the GUB module is $36,000.

MPS III's in-core optimizer, WHIZARD, uses a refined matrix inversion technique and basis update scheme to minimize: fill-in of the basis, growth of the eta file, and the internal representation of the basis inverse. WHIZARD accomplishes the packing of the matrix through a series of checks for redundant portions of the matrix. These redundant portions are systematically removed thereby reducing the original problem. Bradley, Brown, and Graves outline the majority of these reductions and offer results showing the time savings of these approaches [6]. Ketron's president, Mr. Carroll Nelson, estimates that GUB is always preferable to the basic optimizer and preferable to the in-core optimizer WHIZARD in problems where the row/column dimension exceeds 16,384 by 125,000. He also estimates WHIZARD to be superior to GUB until the GUB ratio reaches 80% of the rows or until the maximum dimensions of WHIZARD are reached [37] [27].

3.1.4 AFIT Hardware and Software. The bulk of this research was conducted on the AFIT mainframes, principally on an ELEXSI 6400 multi-coprocessor using the MINOS 5.0 software [34]. MINOS is a general-purpose code for linear and nonlinear optimization. MINOS is a stable, reliable LP package and uses a revised Bartels-Golub update procedure [40:55-69]. By industry standards, MINOS is considered somewhat slow, but it's main drawback for this analysis is the use of a primal simplex implementation that does not create a basis inverse at each pivot of the simplex method.

Therefore, methods that use a basis inverse (GUB,GN,NET) could not be tested locally. Outside assistance for these and other techniques was identified as a necessity at the outset of the research project. This assistance will be mentioned in the appropriate application sections. Each section will report in two parts, the application to TAM and results of the test runs.

3.2 Application of Reformulation to TAM

Much of TAM was written to provide clarity to future maintainers of the model, which is an admirable objective in this day of undocumented models. On closer inspection of the AFCSA formulation, numerous blocks of constraints can be identified that are algebraically repetitive. For example, consider constraint blocks 3.7, 3.8, 3.9, 3.10, and 3.22 and note that all of these constraint blocks could be combined to form two constraint blocks B.4 and B.5 in the revised formulation located in Appendix B. Combining eqn. 3.7 and 3.8 yields:

$$\sum_i \sum_j \sum_n X_{ijklmn} \cdot YLD_{ijklmn} \leq TGT_{klm} \quad \text{for each} \quad k,l,m \tag{3.33}$$

Substituting for TGT_{klm} by replacing with eqn. 3.9 and 3.10 yields two equations:

$$\sum_i \sum_j \sum_n X_{ijklmn} \cdot YLD_{ijklmn} \leq STARTGT_{kl1} \quad \text{for each } k,l,m = 1 \tag{3.34}$$

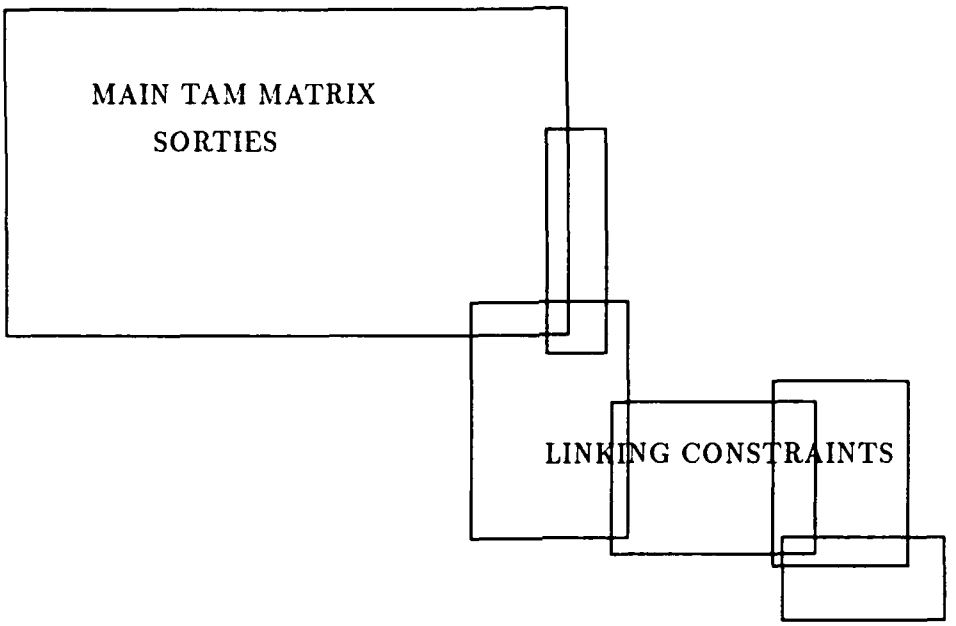
$$\sum_i \sum_J \sum_n X_{ijklmn} \cdot YLD_{ijklmn} \leq TGT_{kl(m-1)} - KILL_{kl(m-1)} + REGEN_{klm}$$
$$\text{for each } k,l,m > 1 \tag{3.35}$$

the right side of eqn. 3.35 can be adjusted by replacing $REGEN_{klm}$ with the contents of eqn 3.22 leaving two constraint blocks B.4 and B.5 in Appendix B.

This situation occurs several times in the current formulation. In fact, the current 32 blocks of constraints can be collapsed into 17 constraint blocks (and three of these are simple bounds) with no loss of purity in the model. MAJ Nick Reybrock of AFCSA concurs with this point and concludes such changes would result in a reduction of 30-45% in the total number of constraints [42]. Since the number of rows is a the major factor in the number of iterations and length of computation time, this change should produce considerable time savings.

Viewing the TAM formulation, several minor adjustments suggest themselves. Referring to Appendix A, the budget awareness factors (BAF in constraint 3.20) adjust the

Figure 3.1. The TAM Matrix



objective function for procuring new aircraft, munitions, or spares but are in the wrong units for the objective function. The terms subtracted from Total Target Value Destroyed (TVD) in the objective function are not target values, but represent some arbitrary small factor applied to new purchases. These terms are similar to penalties applied in goal programming; they are supposed to force the use of current weapon systems prior to purchasing new weapons systems. The objective of TAM analysis is to answer procurement questions, not try to optimize sortie allocation in a war. If the output says buy more F-16s, then let the model output indicate to buy more F-16s. These terms should be dropped from TAM as they force the model to compare apples and oranges.

A big picture of the current matrix of TAM, given in Figure 3.1, reveals a block-angular structure with a single large block and several smaller overlapping blocks appended to the bottom right side. The overlapping blocks contain almost all of the redundant constraint blocks and in a reformulation would become largely extinct.

Other reformulation options include:

1. Reduce the range of options over which a summation occurs. Consider the weather

bands and their use in the model. Currently, they allow TAM to optimize sortie allocation across all weather bands. This is an unrealistic operational goal, since only *GOD* knows the real weather. No commander in the field could expect his staff to perfectly allocate required sorties in this manner. Therefore, if the number of weather bands can be reduced from 6 to 3 then the number of possible variables in TAM can be reduced from 8.75 million to 4.4 million. Or, more realistically, the average weather could be used and the maximum number of variables could be cut to 1.5 million.

2. AFCSA currently uses a deletion program as part of the matrix generator to remove unrealistic aircraft, munition. and target combinations (Reybrock). If this option were expanded, TAM could be recoded to reflect actual combat sortie types. This can be done by removing the sections of code that generate large portions of the matrix, and replacing these sections with a database of the exact sorties we expect to use in a war (e.g. F-111 aircraft would only be matched with current munitions against targets the war plans have them attacking). These changes would provide realistic reductions in the number of variables and generate savings through reduced input/output time of the matrix to the LP package. Note: if the size of TAM can be reduced to remain inside WHIZARD's limits, radical time savings will occur. Major Reybrock estimates the total time to run versions of TAM that fit within WHIZARD is 2–3 hours [41].

A drawback of reformulation is blindly removing portions of TAM's matrix which would effectively eliminate viable alternatives based on operational biases. This will confound the results of a TAM output, but this tends to occur now, albeit on a limited basis, through iteratively running TAM to reach operationally acceptable answers.

Examination of the matrix and the documentation allowed a portion of the redundant constraints to be identified. A manual search of the MPS files (industry standard format for most major LP software packages) showed a capability to remove about one half of the redundant constraints. This amounts to an average reduction of 17% in the number of total rows.

3.2.1 Reformulation Results. Table 3.3 displays sample results comparing solution times of normal TAM problems with reformulated problems after portions of the redundant

structure were removed.

Table 3.3. Reformulation Results

PROBLEM (PCT CHANGE)	ROWS	COLUMNS	DENSITY	SOLUTION TIME	NUMBER OF PIVOTS
PROBLEM.1	91	117	9.58%	16.0s	139
REFORM.1	77	112	11.46%	14.0s	98
	(85%)	(96%)		(88%)	(71%)
PROBLEM.2	180	664	5.67%	34.0s	494
REFORM.2	152	656	6.75%	31.0s	418
	(84%)	(99%)		(93%)	(85%)
PROBLEM.3	269	2076	3.95%	88.0s	941
REFORM.3	227	2064	4.69%	69.0s	647
	(84%)	(99%)		(78%)	(69%)
PROBLEM.4	211	694	4.63%	35.0s	488
REFORM.4	171	678	5.79%	26.0s	296
	(81%)	(98%)		(74%)	(61%)
PROBLEM.5	438	9217	2.43%	527.0s	3051
REFORM.5	371	9197	2.88%	343.0s	1862
	(85%)	(100%)		(65%)	(61%)
PROBLEM.8	420	4844	2.54%	262.0s	1723
REFORM.8	340	4811	3.13%	364.0s	1433
	(81%)	(99%)		(139%)	(83%)
PROBLEM.12	629	15906	1.72%	916.0s	4798
REFORM.12	509	15855	2.13%	762.0s	4568
	(81%)	(100%)		(83%)	(95%)

Several trends are apparent when viewing the results of reformulation:

-The number of rows is reduced 15-20%. Remember, the possibility exists to reduce the number of rows by 35-40%.

-The number of columns is not reduced appreciably in the test problems. The number of columns would not change much, even if all redundant constraints are removed.

-As redundant constraints are removed the density of the problem increases. A problem could be more difficult to solve due to increased overhead associated with increased density, but TAM

problems are very sparse (on the order of 0.1% dense) and this should not be a factor.

Reductions in the number of pivots and in the solution time average about 19-20% overall. In none of the problems did the number of pivots actually increase. A noticeable exception is REFORM.8, with a 139% increase in solution time. The computer system manager suggested this might be a result of specific computer hardware thresholds causing paging back and forth from real memory to disk storage. Monitoring the computer activity during subsequent runs failed to substantiate this theory. However, the solution times for REFORM.8 were erratic compared to all other problems using MINOS on the ELEXSI 6400. This problem could possibly have an ill-behaved matrix or extreme degeneracy problems.

In conclusion, it appears that reformulation of TAM to reduce the number of rows can reduce the solution time. Added benefits to be gained from further reductions of redundant constraints should continue to decrease CPU-time. If values dropped from the formulation are required for analysis reporting, a modification to TAM could handle the required computations. This would be superior to executing several thousand more iterations of the simplex method.

3.3 Application of Crashing an Initial Basis to TAM

AFCSA uses a crashing technique with a sequence of TAM models of increasing size approaching the desired problem size. Using MPS III, a very small problem is solved and the final basis is saved. Then a step up to a slightly larger problem is made and the previously saved basis is used as the initial starting point; in essence, crashing a block of variables into the basis at the very start of the second problem. This procedure is repeated as the problem size is increased to the level dictated by the current analysis project. AFCSA reports this procedure requires 12–24 hours to accomplish with MPS III, but experience shows this is superior to starting TAM from scratch on a large problem and still having no solution in sight after 36–48 hours. Note that MPS III does not require specification of a complete basis, it accepts an incomplete smaller basis as an input and selects the rest of the starting basis on its own [42].

Examining the basis of several small problems made it clear that a particular subset of variables is almost always included. Further research identified several large blocks of columns and rows that were 75-80% basic in all of the small test cases. Preliminary research indicated the possibility to identify 50-70% of the basic columns and rows prior to solution. Given the possibility of starting at an infeasible extreme point, it looked promising to identify a large portion of TAM's optimal basic solution.

MINOS, however, was not as capable as MPS III in this regard. MINOS required identification of an exact set of basic columns and rows. It would not accept a partial list and identify the remaining 30-50% of the basis required. MPS III accepts a subset from a previous smaller problem and selects further entries intended to reduce infeasibilities. Using MINOS, experiments were conducted with the most promising subsets and a best guess from the remaining available choices. A dozen attempts were made with different combinations of promising blocks of columns and rows. In each case the rest of the basis set was completed with a set of best guesses.

3.3.1 Crashing Results. The preliminary runs with this technique were plagued with errors indicating the matrix was singular and no attempt was made at solution by MINOS. In adjusting to a strategy that provided MINOS a complete non-singular basis, the experimenter was forced away from the most promising combinations of rows and columns to a strategy that guaranteed only 40-50% of all possible basic entries. Still, this should have provided enormous benefit.

MINOS accepted the initial basis in these cases, but took longer both in terms of time and number of iterations to run to an optimal solution. The final runs were a mixture of pressing singular errors and trying to get a large percentage of good prospects for the final basis. The experimenter succeeded in reducing the time and iterations only once. The reduction was not substantial; approximately 1.5% of the total CPU time and 7 iterations on a problem requiring 140 pivots.

A capability that MINOS does have is to select an all slack basis as a starting point. This was attempted with each of the test problems and the results showed a general reduction in CPU-time of 20-25%. This suggests that a thorough knowledge of the problem

matrix and a clear idea of software capabilities can allow "tuning" the algorithm to suit specific needs.

It appears through communication with AFCSA that due to the added capability with MPS III, basis crashing still offers some hope of reducing the run time. They are continuing to attempt to apply this technique on some of their larger problems based on this analysis (Reybrock2). A final note: many of the rows and columns identified as generally basic are located in the redundant constraint section and would be removed by reformulation.

3.4 Application of the Simplex Method for Bounded Variables

Most modern LP packages have the capability to exploit upper bounds, lower bounds, and ranges for variables (certainly both MINOS and MPS III do). TAM has a series of explicit constraints which are actually simple upper and lower bounded variables. These constraints are easily identified as rows with a single non-zero element. The lower bounds are always zero in the "small theater war" version of TAM and the simple upper bounds were selected as an opportunity to test this method.

A computer program was written to check the TAM matrix for rows containing a single non-zero coefficient. These rows were checked for removal from the TAM matrix and entrance into the special format for upper and lower bounded variables in MINOS. The appropriate rows were then removed from the TAM matrix and the bounds section was modified accordingly.

Because of the very short CPU-time required to alter the TAM matrix using this technique, no statistics were taken on the length of time to locate and modify the matrix. It is assumed that if this technique shows promise, the TAM pre-and post-processors can be altered and the matrix generator would simply write these items to the bounds section instead of the coefficients section of the MPS file. There would then be no appreciable cost in CPU-time due to removing the simple upper and lower bounds compared with the current method used. MPS III and WHIZARD have pre-reduce features that may automatically identify simple bounds, but it is always better for the modeler to do this

work himself.

3.4.1 SUB Results. At the Naval Postgraduate School, sample TAM problems were tested for SUB and GUB structure. The search for rows meeting either condition was conducted as part of the GUB tests mentioned in the following section. Table 3.4 outlines the results of examining TAM for SUB and GUB structures.

Table 3.4. Results Of Heuristic Search for SUB and GUB Rows

PROBLEM NUMBER	ROWS	COLUMNS	SUB ROWS	GUB ROWS	UNSCALED GUB ROWS
PROBLEM.1	91	117	10	22	47
			10.9%	24.2%	51.6%
PROBLEM.2	180	664	20	44	92
			11.1%	24.4%	51.1%
PROBLEM.3	269	2076	30	66	138
			11.2%	24.5%	51.3%
PROBLEM.4	211	694	24	60	75
			11.3%	28.4%	35.5%
PROBLEM.5	438	9217	47	107	110
			10.7%	24.4%	25.1%
PROBLEM.8	420	4844	48	121	150
			11.4%	28.8%	35.7%
PROBLEM.12	629	15906	72	181	225
			11.4%	28.8%	35.8%

Reviewing the tests for SUB and GUB structure a few points are noteworthy:

-the percentage of SUB rows holds nearly constant as the problems increase in size and complexity. The percent of SUB rows is constant at about 11%.

-the percent of GUB rows within the matrix also holds nearly constant between 25-29%, with scaling this appears to reach a maximum of 36%.

A benefit from removing SUB rows is the additional efficiency gained by the increased percentage in GUB structure. The SUB rows tend to be part of the GUB structure of the

original problem. Their removal allows the heuristic search for GUB rows to identify another set of rows for GUB operation. In all cases tested this would result in the same number of rows available for GUB structure as before, but now in a matrix reduced in size. For instance in PROBLEM.1 there are 91 rows. If we remove the 10 SUB rows from the structure and reapply the heuristic search for GUB structure we again find 22 rows with the appropriate structure. Now the 22 GUB rows form 27% of the remaining matrix instead of 24% of the original structure. The benefits of GUB structures within the TAM matrix will be covered in the next section.

The TAM sample problems were originally run on MINOS with the straight simplex application. The TAM sample problems were then adjusted for the SUBs and rerun on MINOS using the simplex method. This adjustment reduced the number of rows but did not change the number of columns. The density of the matrix was increased in each problem in the SUB runs. For the smallest problem the increase in density was as large as 1.0%. For the largest problem in the SUB run the increase in density was 0.33%. Table 3.5 displays the results of the SUB tests.

Table 3.5. SUB Test Results

PROBLEM	SIMPLEX CPU-TIME	PIVOTS	SUB ROWS	SUB CPU-TIME	PIVOTS
PROBLEM.1	16.0s	131	10	15.0s	107
			10.9%	94%	82%
PROBLEM.2	34.0s	494	20	33.0s	523
			11.1%	97%	106%
PROBLEM.3	88.0s	941	30	67.0s	610
			11.2%	76%	65%
PROBLEM.4	35.0s	488	24	32.0s	407
			11.4%	91%	83%
PROBLEM.5	527.0s	3051	47	369.0s	2136
			10.7%	70%	70%
PROBLEM.8	262.0s	1619	48	235.0s	1714
			11.4%	90%	99%
PROBLEM.12	916.0s	4798	73	882.0s	5178
			11.4%	96%	108%

The results from the SUB tests have some slight inconsistencies but suggest some general trends. SUB reduces the solution time but the variance in reductions is large and erratic. Most of the reductions are on the order of 4% with an occasional large reduction of around 27%. In some cases the number of actual pivots performed increased slightly. The data suggests that when the number of iterations went down, a significant reduction in time occurred; and when the change in iterations was insignificant or went up slightly, overall reduction in solution time was insignificant with SUB.

3.5 Application of GUB and NET to TAM

The staff of the Naval Postgraduate School tested TAM for a GUB/GN/NET structure, using the X-System (copyright 1989 INSIGHT,INC.) software package on an IBM 3033/AP mainframe under CP/CMS. TAM was also subjected to a simple automatic cascading solution technique (e.g. Brown and Graves 1987). The key issues in using these applications are obtaining a maximum GUB/GN/NET set in the TAM matrix and, if it is sufficiently large, testing TAM for faster solution through the appropriate method.

Brown and Thomen conducted extensive research in the area of identifying maximum GUB sets. They conclude that finding guaranteed maximum GUB sets is an NP-complete problem, and therefore, use heuristic techniques to quickly find near- maximum sets. The heuristic searches are polynomial-time algorithms and they report good success using these techniques with the X-System software to handle GUB and similar problems (GN, NET, etc...) [9].

The tests combined the GUB identification heuristic, a SUB identification algorithm, and a GUB solver for the LP. It is important to note that any improvements in solution time must include the necessary CPU-time required to isolate a GUB structure, identify the GUB rows, communicate this data to the GUB software, as well as actual solution time via GUB. Benchmark simplex solution times were accomplished with a simplex algorithm of advanced design in the X-System software. As mentioned, the GN/NET tests are described by Olson [38].

3.5.1 GUB and NET Results. The results of the GUB/NET tests on sample TAM problems are in Table 3.6. The first three problems used the two-pass GUB search while the last four problems used a single-pass heuristic. The results reported on the last four problems are for GUB sets slightly less than the values reported in Table 3.4. Respectively the GUB row counts for PROBLEMs 4-12 are 59, 102, 118, and 177.

Table 3.6. GUB Test Results

NAME	SIMPLEX		GUB			NET		
	TIME	PIVOTS	TIME	PCT.	PIVOTS	TIME	PCT.	PIVOTS
PRO.1	.8s	123	.6s	75.0%	111	.9s	112.5%	120
PRO.2	2.5s	175	2.1s	84.0%	183	2.4s	96.0%	157
PRO.3	21.3s	522	15.4s	72.3%	450	13.1s	61.5%	434
PRO.4	1.7s	230	1.2s	70.5%	196	1.6s	94.1%	228
PRO.5	308s	1801	280s	90.4%	1514	311s	101%	1696
PRO.8	137s	1965	85s	62.0%	1657	87s	63.5%	1204
PRO.12	561s	3233	525s	93.6%	3031	503s	89.7%	2991

A few obvious items from the results of the GUB/NET tests are:

-in each case the time for solution using the GUB technique was shorter (ranging from 6% to 38%), while the NET technique often was shorter it seldom outperformed GUB,

-the number of total iterations using GUB was generally less by 10-15%,

-in PROBLEM.2 the number of pivots actually went up but the total time required for solution was reduced by about 10%. This is a case illustrating more numerous pivots that actually require less computational effort,

-the scaling technique was not used which could have identified a GUB set of 35% of the total number of rows, as opposed to the 25% of total rows shown in Table 3.4.

A note to this portion of the research is the effect of reformulation on the use of SUB and GUB. Reformulation of TAM could affect SUB and GUB techniques by reducing the rows used for both GUB and the SUB methods. In doing this, the matrix becomes more dense compensating for the removal of the redundant constraints. Therefore this technique eliminates the possibility of using SUB techniques and could reduce the effects of a GUB strategy. The GUB heuristic would be able to identify new GUB structures, but the size of the GUB set relative to the new matrix is an open question. Further research into this possibility would be required. As GUB was generally preferable to NET, NET will be discarded as a solution to this problem though it's tendency to reduce TAM's CPU-time is noted.

3.6 Application of Karmarkar's Algorithm to TAM

Military Airlift Command (MAC) Headquarters tested TAM using Karmarkar's method implemented on the newly-installed AT&T KORBX system. This system is basically a state of the art marriage of specific hardware and software to form a "more perfect union." The KORBX machine is a mini- supercomputer with 14 co-processors running under UNIX.

The application of Karmarkar's algorithm is a blend of several techniques and options. The KORBX system contains four solvers that can be selected based on knowledge of the specific problem and contains several options for tuning the algorithm to optimize solution speed. The separate solvers are the Primal, Dual, Primal-Dual, and Power Series. As previously mentioned, KORBX checks the primal-dual feasibility gap to determine optimality. At this time no simplex algorithm is run on KORBX and a direct comparison of simplex and Karmarkar's algorithm cannot be made.

3.6.1 KORBX Test Results. The test results from KORBX are located in Table 3.7. The raw results for all solver methods using KORBX are displayed.

Reviewing the results a few key points are:

-in several instances the KORBX system did not reach optimality, these cases are identified by having reached an upper limit of 200

Table 3.7. KORBX Test Results

NAME	PRIMAL		DUAL		PRIMAL-DUAL		POWER SERIES	
	PIVOTS	TIME	PIVOTS	TIME	PIVOTS	TIME	PIVOTS	TIME
PRO.1	24	3.0s	26	8.6s	27	3.2s	12	2.3s
PRO.2	29	8.6s	30	7.4s	40	11.4s	21	8.3s
PRO.3	41	22.1s	45	22.3s	64	39.7s	30	26.9s
PRO.4	29	8.5s	31	7.8s	33	9.9s	20	8.3s
PRO.5	66	107.9s	200	240s	107	223.4s	200	597.5s
PRO.8	115	101.1s	200	137s	95	114.9s	51	91.4s
PRO.12	122	313.8s	200	424s	157	555.7s	200	1078s

iterations,

-as the problem size increased the PRIMAL solver was the more effective and more stable in terms of reaching an optimal solution. This makes sense as TAM has a large number of columns and the column section grows rapidly with increased problem size,

-the number of iterations is not comparable with those in the simplex solution techniques,

-in an absolute sense, this method was quicker than the previously obtained simplex solutions on the larger problems, and slower on the smaller test cases.

Future references to this method will address the PRIMAL solution technique only. This is logical since TAM's matrix structure ensures a primal technique will be more efficient than any dual technique.

A final point– note that no simplex benchmark was available for this technique. The KORBX system does not contain a simplex- based method and there are no plans to incorporate one. To adequately resolve which advanced technique is inherently quicker, this issue must be addressed. Chapter 4 deals with this problem.

3.7 Application of the Folklore Approach

The Folklore method was tested due to its novelty and because the TAM matrix has a significant number of zeros in its right-hand side. The TAM matrix was adjusted with a new variable added, its value fixed equal to 1, and this variable added to the rows with a zero right-hand side. Use of this technique does not require any further adjustment to TAM.

Table 3.8. Folklore Test Results

PROBLEM	SIMPLEX TIME	PIVOTS	FOLK. TIME	PCT. CHG.	PIVOTS	PCT. CHG.
PROBLEM.1	16s	139	16s	100%	144	100%
PROBLEM.2	34s	494	31s	91%	388	79%
PROBLEM.3	88s	941	78s	89%	792	84%
PROBLEM.4	35s	488	37s	106%	527	108%
PROBLEM.5	527s	3051	449s	85%	2537	83%
PROBLEM.8	262s	1722	252s	96%	1683	98%
PROBLEM.12	916s	4798	982s	107%	5135	107%

The results from the folklore method in Table 3.8 are inconclusive with no real hope for great gains as suggested in Charnes' report. The method reduced the total time significantly only one time; generally the method shows no improvement, and often shows an increase in solution time. Therefore it can be ruled out as a possibility.

3.8 Summary

A review of the test results show that the following methods can be rated as promising: reformulation, SUB, GUB, and KORBX. The following methods require further testing: basis crashing (as a tuning technique with MPS III), GN, cascading, and Marsten's method. The results obtained from the reported method from LP folklore suggest that it can be discarded as an option, and since NET shows no improvement over GUB it also is discarded.

IV. COMPARISONS

4.1 Introduction

To evaluate which technique is fastest, a comparison needs to be drawn between the KORBX system using Karmarkar's algorithm and the simplex-based methods. As mentioned, in an ideal world this testing would be accomplished under similar conditions on the same computer hardware. As this was not the case, two avenues of comparison present themselves: an absolute comparison without regard to computer architecture and capability or a relative comparison that removes the confounding effect of the computer architecture and capability.

4.2 Absolute Reference

One way to compare dissimilar algorithms executed on different computer hardware is direct examination of the total CPU-time taken to reach termination. A direct coparison of the three systems using selected techniques are compiled in Table 4.1. This comparison reveals that on the smaller problems the X-System software using GUB (on an IBM/3033 mainframe) was the quickest while on the larger problems the KORBX system using Karmarkar's algorithm was the fastest. A break point seems to occur around the relative size of a problem similar to PROBLEM.8.

Table 4.1. Comparison with an Absolute Reference

PROBLEM	KORBX	X-SYSTEM		MINOS	
NAME	PRIMAL	SIMPLEX	GUB	SIMPLEX	SUB
PRO.1	3.0s	.8s	.6s	16s	15s
PRO.2	8.6s	2.5s	2.1s	34s	33s
PRO.3	22.1s	21.3s	15.4s	88s	67s
PRO.4	8.5s	1.7s	1.2s	35s	32s
PRO.5	107.9s	308s	280s	527s	369s
PRO.8	101.1s	137s	85s	262s	235s
PRO.12	313.8s	561s	525s	916s	882s

In no case was the MINOS software faster, and in most cases it compared very poorly with the winner. MINOS ranged from 3 to 20 times slower than the fastest implementation. This occurred consistently no matter what technique was applied with MINOS.

The data seem to suggest that as the problem size grows, the KLP-solver on the KORBX system reaches a point where it crosses from being rather inefficient to being more efficient. In the largest problems the increase in speed was between 50–60%. Figure 4.1 displays the concept in visual terms.

The data does not prove the AT&T implementation will continue to grow in efficiency relative to simplex-based methods. Both the X-System and MINOS recover on PROBLEM.12 and close the gap created on PROBLEM.5; initially they were slower by 3 to 5 times on PROBLEM.5 and they are only 1.6 to 3 times as slow on the larger problem. Further testing would be required to verify this situation. In any case, it does seem that as problem size increases, the KORBX system provides the faster solution technique.

4.3 Relative Comparisons

To adequately examine the respective efficiency of an algorithm, testing should be conducted on the same machines under similar conditions. Due to the wide sweep of this analysis, that was impossible. As mentioned, no simplex implementation is on a KORBX machine, nor is any proposed. Nevertheless an attempt was made to remove the effects of hardware and examine the efficiency of respective algorithms apart from the machines on which they run.

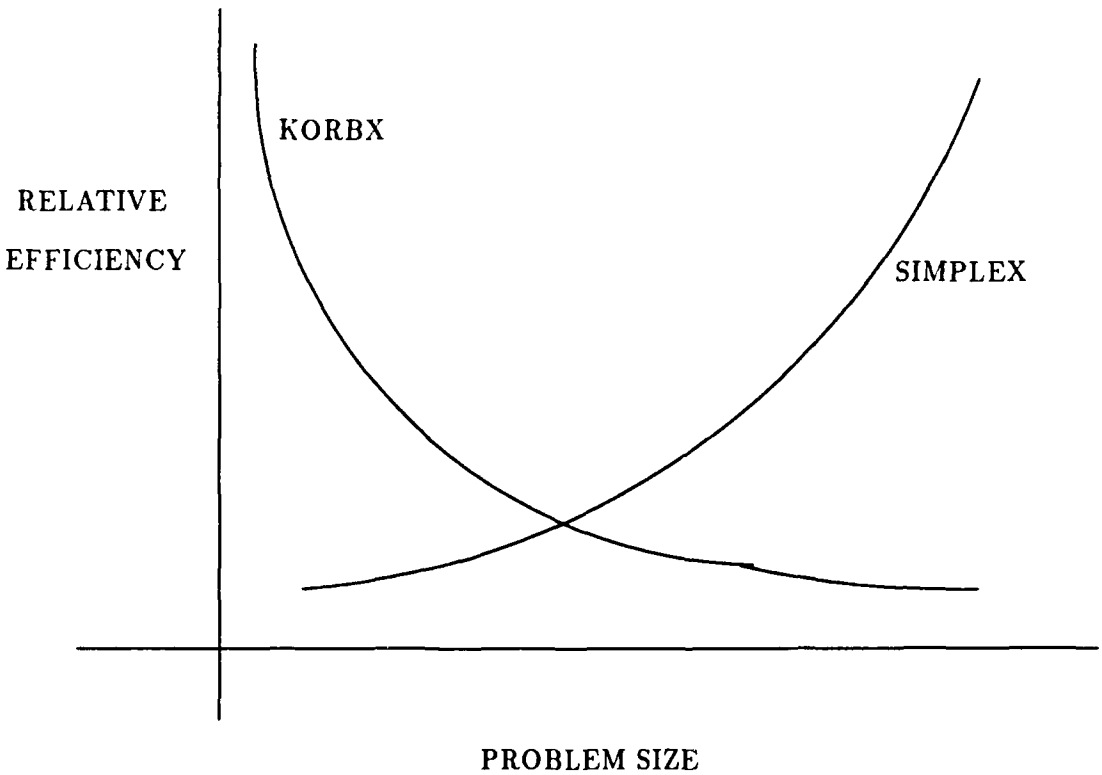
Many different techniques are used to measure the respective speed of different computers. One need only listen to any vendor to hear numerous statistics thrown about that will indicate the various benefits of their own machines in terms of performance. Several good measures are likely candidates:

MIPS- *m*illions of *i*nstructions executed *p*er *s*econd,

MFLOPS- *m*illions of *f*loating *p*oint *o*perations performed *p*er *s*econd,

VAXMIPS- a number representing the relative speed a machine has with respect to a VAX. For instance if a machine is rated at

Figure 4.1. Hypothetical Efficiency of Karmarkar's Algorithm



3 VAXMIPS this indicates it can process a million instructions 3 times as fast as a standard VAX.

WHETSTONES- a comparison of speed operating on a series of standard problems. This measure has lost much of it's allure as some manufacturers have optimized their hardware for this test and it is no longer considered a reliable measure.

Whetstones and MIPS were discarded as possibilities since more data was available on the other measures. In the case of KORBX and the IBM/3033 at the NPS information on VAXMIPS was also not available. Discussions with the system managers at all sites confirmed a rule-of-thumb in the industry for computing VAXMIPS based on 75% of the vendors' MIPS rating. This measure is often inflated or represents peak efficiency rather than standard operating capability. The KORBX machine is rated at 40 MIPS and therefore was assigned a value of 30 VAXMIPS. The IBM/3033 at the NPS is rated at 4.7 MIPS and was assigned a value of 3.5 VAXMIPS. The ELEXI 6400 at AFIT is rated at 7 VAXMIPS.

MFLOPS was selected as a second criteria based on it's availability from all sources. The KORBX is rated at 94 MFLOPS, the IBM/3033 is rated at 10-12 MFLOPS, and the ELEXI 6400 is rated at 2 MFLOPS. Being slightly skeptical of these measures, a small test was run at AFIT using the TAM sample problems to compare the ELEXI and a VAX 11/785 (the VAX11/785 is rated at 1 VAXMIP and 1 MFLOPS). The VAXMIP rating reflected correct speed relationships between the two machines, but the MFLOPS rating for the ELEXI appeared to be grossly in error. The ELEXI MFLOPS rating was monitored throughout the comparisons of the algorithms. In all cases, the ELEXI MFLOPS rating distorted the comparisons which, otherwise, were very stable. The calculations indicated that MINOS was faster than the competition by some 20–30 times. Since this result is totally unbeleivable, the ELEXI/MINOS MFLOPS ratings are not included.

The results in Table 4.2 were computed by setting the KORBX system as the standard; that is a comparison was made between the speed rating of each machine relative to the KORBX. A factor was then applied to the results from the X- System and MINOS to

compensate them for operating on slower hardware. For instance, the VAXMIPS for the KORBX are 30, the VAXMIPS for the IBM/3033 are 7, the rating of the IBM was divided by the rating for the KORBX, and this factor was applied to the X-System results to show a relative comparison. The factors applied in each case were:

- X-System factor = 3.5/30 = .117 for VAXMIPS
- X-System factor = 12/94 = .127 for MFLOPS
- MINOS factor = 7/30 = .233 for VAXMIPS, and the MFLOPS rating was not used.

The results of this analysis are displayed in Table 4.2. In each case the first line in the chart reflects the VAXMIP comparison for a specific problem, and the second line reflects the MFLOPS comparison.

Table 4.2. Comparison with a Relative Reference

PROBLEM NAME	KORBX PRIMAL	X-SYSTEM SIMPLEX	X-SYSTEM GUB	MINOS SIMPLEX	MINOS SUB
PRO.1	3.0	.09	.07	3.7	3.5
	3.0	.10	.08		
PRO.2	8.6	.58	.25	7.9	7.7
	8.6	.32	.27		
PRO.3	22.1	2.49	1.80	20.6	15.7
	22.1	2.72	1.97		
PRO.4	8.5	.20	.14	8.1	7.4
	8.5	.22	.15		
PRO.5	107.9	35.9	32.7	123.0	86.1
	107.9	39.3	35.7		
PRO.8	101.1	16.0	9.9	61.3	54.8
	101.1	17.5	10.9		
PRO.12	313.8	65.5	61.3	213.7	205.8
	313.8	71.6	67.0		

In light of these results (remembering the lowest number across any row indicates the hypothetically fastest algorithm) the KORBX implementation of Karmarkar's algorithm fares less well. In general, MINOS keeps pace with it, and the X-System appears to be

5–20 times faster on the smaller problems and 3–5 times faster on the larger problems. These results suggest two posibilities; the first is that the problems selected are too small to adequately test Karmarkar's algorithm and the KORBX system. The test cases force KORBX to operate in a region were it cannot display a speed advantage and, therefore, the simplex-based methods appear more efficient. The other possibility is that simplex-based methods are more efficient than Karmarkar's algorithm, and if they were coded for a supercomputer with the advantage of parallel-vector processing they would show even faster absolute times than Karmarkar's algorithm and the KORBX system.

4.4 Summary

The KORBX system and Karmarkar's method, while slower on small problems, appears faster than the simplex algorithms tested on their respective hardware. When respective hardware is taken into account, the simplex-based methods are faster than Karmarkar's algorithm on problems of this size. Further tests should be run by AFCSA testing larger classified problems on the IBM/3084 at the Pentagon vs. the KORBX system at MAC to definitely answer whether problem size is an issue.

V. RECOMMENDATIONS

5.1 Summary

A recap of the first four chapters shows that there are many ways to implement Linear Programming. A taxonomy has been presented to classify the major techniques according to similar characteristics. Major categories included special structures and factorization, interior point algorithms, and restructuring techniques. Within these categories, a number of techniques have been considered for reducing TAM's CPU-time. These included basis crashing, Dantzig-Wolfe decomposition, cascades, GN, NET, SUB, GUB, Khacian's ellipsoid method, Karmarkar's algorithm, Marsten's dual-affine interior point method, reformulation, goal programming, and the folklore approach.

A number of these techniques were not selected for application to TAM due to poor prospects of reducing the CPU-time, these included: Dantzig-Wolfe decomposition, Khacian's ellipsoid method, and goal programming. Some techniques could not be tested due to logistical problems, among them are: GN, cascades and Marsten' method. The NET and LP folklore method test results suggested they could be discarded as considerations. The remaining techniques either reduced CPU-time when tested or require further research (basis crashing).

An examination of these techniques revealed that the KORBX implementation of Karmarkar's algorithm was fastest on large problems, while the X-System using the GUB technique was fastest on smaller problems. A relative comparison between methods considering the effect of computer hardware suggests that simplex-based methods are faster than Karmarkar's interior point method.

Given these details, the remainder of the research is devoted to identifing the best alternatives for AFCSA to reduce TAM's CPU-time.

5.2 Computer Time ($ per CPU-minute)

The Pentagon estimates the value of IBM/3084 CPU-time at $420.00 per hour. If TAM is run 500 times each year, with each run taking 20–200 CPU minutes (the mean

time being approximately 2 hours), then a rough estimate of the money spent to run TAM is $500 \cdot 2 \cdot \$420 = \$420,000$ for computer time each year. Given the 25% criteria at the start of this research, we are then trying to save AFCSA about $105,000 of computer time. These calculations will be useful for judging possible alternatives for saving money and time.

5.2.1 KORBX at MAC. As this capability resides only at Scott AFB, Illinois it would require considerable travel on an annual basis. If TAM needed to be run 20 times each for 25 analysis projects throughout the year, this would require 25 trips to MAC/HQ. Assuming the details of occupying the KORBX system for a solid week could be worked out, the approximate cost to send a officer TDY at $1000 a trip for a 5-day week would cost $25,000 in TDY funds. If the KORBX system could solve TAM in one-third the time taken at AFCSA, then a net savings of $(.33 \cdot \$420,000) - \$25,000 = \$114,000$ would be realized. Note this does not include the time to generate the problems, download those problems to a disk drive, and cut a tape. These items, though not considered, are by no means trivial or without cost.

Several other problems exist with using KORBX to solve TAM. Computer time savings are realized at AFCSA but at a cost to MAC/HQ. The question arises whether this is a savings to anyone (i.e. the USAF) or merely creative bookkeeping. Another difficulty is that every agency operating TAM would be in the same position of migrating to MAC whenever they needed to run TAM. Finally, an additional 25 weeks TDY on a 5-7 man Air Force office each year would be a hardship. A secure high-speed phone capability would alleviate the last two difficulties; however, no modem capability exists at this time, and none is foreseen in the future. Given these drawbacks, this option is best viewed as an interim measure due to several points suggesting it could be unreasonable. If the high speed modem capability could be worked out, then this alternative becomes more attractive.

5.2.2 MPS III and GUB/GN/NET. This option would require the outlay of an additional $36,000 to purchase a GUB module, that according to Ketron, will not reduce TAM's CPU-time. TAM, as currently formulated, has a GUB set of 25-28% of the available

rows, and the MPS III GUB module requires a problem with an 80% GUB set to show a time benefit. Since this technique would cost extra money and not produce results it is discarded as unreasonable.

Since no commercial code exists at this time to take advantage of GN/NET sets, and TAM's current formulation does not provide a substantial GN/NET set these possibilities are also discarded.

5.2.3 Basis Crashing. Though not a viable technique with MINOS, basis crashing could be a possibility with MPS III. Mainly this would be a function of tuning the algorithm's parameters to take advantage of TAM's special structure (i.e. starting with an all slack basis, using partial pricing to reduce the column search space for entering basic variables and reducing the number of degenerate pivots, applying any scaling features, and adjusting the feasibility tolerances for entering variables and for optimality). Application of these tuning features, to effectively crash the basis, would be algorithm-specific and would need to be done by personnel familiar with SCICONICS and its specific tuning features.

This technique would require no extra funds, but would require further experimentation with TAM on MPS III. There is no way to predict how effective crashing techniques would be and it is not considered in the recommendations pending further testing.

5.2.4 SUB. Estimates suggest that an excellent FORTRAN programmer would need a month to implement SUB with the current TAM code. This would require a search for SUB rows prior to matrix input to the solver and flagging these items so that MPS III will handle these variables as simple bounds. MPS III during the pre- solve phase may identify these items, but it would save some effort if these items were coded as part of the model. If MPS III does not identify these rows itself, then a time reduction can be expected (10% in the tests with MINOS). The 10% expected savings translates to a savings of $42,000 in computer time each year at AFCSA. This does not approach the goal of $114,000 but for a minimum of effort it seems a significant return for good modeling practice. A similar savings could be expected with SCICONICS at the other agencies.

5.2.5 Reformulation. This method offers the most hope of reduced time savings for minimum dollar expense. Estimates suggest it would require two excellent FORTRAN programmers and an OR analyst a year to recode TAM. Model development could be accelerated through use of newer matrix generation techniques. The redundant portions of the matrix could be dropped, the SUB rows could be identified, and TAM could be restructured to take advantage of specific tuning capabilities of MPS III and SCICONICS.

Possibly the biggest benefits would come from reducing the TAM model so that it fit within the constraints of WHIZARD. AFCSA analysts acknowledge TAM runs in 2–3 hours for models that fit within WHIZARD's constraints. This translates to solving TAM in 1/6 the normal time requirements, or, a savings of $350,000 in computer time each year.

This could be done with TAM by reducing the number of weather bands used within the LP from 6 to 3 or even to 1. Using 3 weather bands instead of 6 reduces the normal TAM scenario from a model that is 3500x250,000 to a model that is 3500x125,000, a size that would just fit within WHIZARD. If the average weather were used across the entire model then the number of variables would be reduced to 1/6 the current number. The average TAM matrix would then contain about 45,000 variables, and have room to expand when the proposed ECM and airbase operability features are added.

5.3 Final Recommendations

The recommended courses of action in ranked order are:

-above all else, reformulate with a view to reducing redundancies, reducing the model size, and using SUBs,

-flagging the SUBs, it cost very little to do this,

-interim consideration of travel to and from MAC/HQ to utilize the KORBX system,

-spending money for new GUB software with little chance of benefit,

-spending big money for new hardware. If you duplicate MAC's $5-6 million purchase, then you should expect speed .

-verdicts are still not in on a few longshots: basis crashing, GN, cascades, and Marsten's method.

Appendix A. *AFCSA Formulation of TAM*

A.1 Introduction

The Theater Attack Model (TAM) is a large scale linear programming (LP) model designed to run using commercially available LP software packages. TAM was developed as an analytical tool for determining the impacts of budget, attrition, force structure, targeting decisions, and munitions inventories on warfighting capablility in a theater scenario. To perform the MAST (Munitions, Aircraft, Spares Trade-off) study the model was modified to allow the inclusion of spares supportability for sorties flown. In this configuration the model is capable of providing insights into the best allocation of additional budget dollars for procurement of aircraft, spares and munitions to enhance our capability to destroy targets given appropriate limitations on the above resources.

A.2 Formulation

A.2.1 Subscripts. The subscripts utilized in the formulation descriptions are as follows:

i = aircraft type. Aircraft types may be entered with each type being either a new/different aircraft or a special capability on an existing aircraft, e.g. F-16's with wild weasel capability.

j = munition type.

k = target type. Targets are distributed across the available distance bands.

l = distance band. Distances are measured in nautical miles starting at the political border. Targets do not move between distance bands during the war. Distance is used to capture aircraft range differences, standoff weapons used as range extenders, target spatial distributions, and variations in target value based upon positioning. Four distance bands were used in the MAST formulation of the model. Due to the specific model formulation, more than four distance bands are difficult to implement.

m = time bands. Time is measured in days with attrition and target values varying between time bands. The MAST formulation used three time bands. More than four time bands requires extensive modification of the model.

n = weather bands. Each weather band is described by cloud ceiling and visibility. The best delivery profile for each weather band is selected based on the aircraft, munition, target combinations. Expected attrition and effectiveness are calculated based on the delivery profile selected. These calculations occur off-line and are an input to TAM.

p = spares resource. Different resource types are available for each A/C. Cost is the main driver for the different resources. Spares resources are identified as an expected useage rate per sortie flown. Spares inventories represent the expected number of sorties supported by spares.

A.2.2 Objective function.

$$\text{Maximize} \quad TVD = \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n X_{ijklmn} \cdot YLD_{ijklmn} \cdot TGTVAL_{klm} - BAF \quad (A.0)$$

Where: TVD = target value destroyed. Total value of all targets destroyed as a result of the optimization.

X_{ijklmn} = sorties. The X_{ijklmn} are the primary decision variables, each sortie is identified as a specific mission consisting of aircraft i carrying munition j against target k located in distance band l during time m and weather condition n.

YLD_{ijklmn} = yield (fractional target kills) per sortie, calculated based on the expected mission effectiveness and attrition.

$TGTVAL_{klm}$ = target value, assigned by target type, time period, and distance band. Used by TAM to prioritize targets attacked, TGTVAL represents the value of one target of type k at distance l and time m.

BAF = buget awareness factor, a small adustment to the objective function which forces sensitivity to existing inventories of munitions and aircraft during the optimization. See constraint 19.

A.2.3 Constraints The Theater Attack Model was developed to provide an alternative methodology for the determination of munition requirements. However, the model also has the flexibility to address other analysis issues, such as; trade-offs between dollars spent on force structure, spares, and munitions; attrition management; and capability assessment.

The constraints imposed on the objective function provide the flexibility which allows the analyst to customize the model formulation for each analysis task. The contraints currently available are detailed below. Among these constraints are a set of "core" constraints which maintain the validity of the model. The "core" constraints are identified with an asterisk and must be used in any formulation. To enhance the ease of use of the constraints in various formulations each constraint is numbered and selectable from the Job Control Language. The numbers in parentheses are associated with this JCL selection table.

$$\sum_j \sum_k \sum_l \sum_n \frac{X_{ijklmn}}{TS_{ijklmn}} = EAC_{im} + NAC_{im} \qquad \text{for each } i, m \qquad \text{(A.1)}$$

For each aircraft type this constraint ensures that the total aircraft used during each time period equals the sum of existing aircraft used plus any new aircraft purchased.

TS_{ijklmn} = maximum number of sorties that could be flown by one type i aircraft during time period m against target k using munition j in weather band n to a penetration depth l.

EAC_{im} = the number of existing type i aircraft used during time period m. The available pool of existing aircraft consists of starting aircraft (those in place on D-Day) plus filler aircraft from the previous and current time periods, plus new aircraft purchased during previous time periods, minus aircraft which have been attrited during previous time periods. (See constraint 3.)

NAC_{im} = the number of new type i aircraft purchased for use starting in time period m.

$$EAC_{i1} \leq STARTAC_i \cdot FILL_{i1} \quad \text{for each } i \qquad \text{(A.2)}$$

For each aircraft type, this constraint ensures the total existing aircraft used during time period one does not exceed the aircraft stationed in theater on D-day plus any fillers which arrive during this time period.

$STARTAC_i$ = the number of starting aircraft of type i.

$FILL_{il}$ = the number of type i aircraft arriving during time period one. Filler aircraft are available for use during the entire time period, therefore if the aircraft are scheduled to arrive part way through the time period a reduced number of fillers are added during the current time period and the remainder in the following time period.

$$EAC_{im} \leq STARTAC_i + \sum_m FILL_{im} + \sum_{m-1} NAC_{im} - \sum_{m-1} ATTRIT_{i,m} \quad \text{for each } i, m, n \tag{A.3}$$

For each aircraft type, this constraint ensures the total existing aircraft used during a time period does not exceed the aircraft available at the beginning of the time period plus any fillers which arrive during the time period. For example, aircraft available at the beginning of time period two are equal to starting aircraft plus time periods one and two fillers plus time period one new purchases minus aircraft attrited during time period one.

$ATTRIT_{im}$ = = the number of type i aircraft attrited during time period m.

$$\sum_j \sum_k \sum_l \frac{X_{ijklmn}}{TS_{ijklmn}} \leq PRCNT_n \cdot \left[STARTAC_i + \sum_m FILL_{im} + \sum_m NAC_{im} - \sum_{m-1} ATTRIT_{im} \right] \quad \text{for each } i, m, n \tag{A.4}$$

For each aircraft type, this constraint ensures the aircraft usage across the weather bands is consistent with the expected weather distribution. This constraint can cause the problem to go infeasible if the mission file does not contain missions with effectiveness greater than zero for each aircraft type and weather band. This can occur due to operational constraints (ie. an aircraft type that is not used during bad weather). To regain feasibility it is necessary to add a dummy mission to the mission file. The dummy

mission should have low effectiveness in the necessary weather band, no attrition, and carry a dummy munition. (If constraint number 23 is used, then the appropriate weather constraint is number 24.)

$PRCNT_n$ = the percent of the time that type n weather is expected to occur.

$$\sum_i \sum_k \sum_l \sum_m \sum_n X_{ijklmn} \cdot LOAD_{ijl} = EMN_j + NMN_j \qquad \text{for each } j \qquad \text{(A.5)}$$

The total munitions used throughout the war must equal the sum of the sorties flown with that type munition times the load (number of munitions carried on that sortie). The loads change with aircraft type and distance to the target.

$LOAD_{ijl}$ = the number of type j munitions carried on aircraft type i to distance band l.

EMN_j = the number of existing inventory munitions used.

NMN_j = the number of additional type j munitions purchased.

$$EMN_j \leq STARTMN_j \qquad \text{each } j \qquad \text{(A.6)}$$

Limits the number of existing munitions used to the starting inventory.

$STARTMN_j$ = the starting inventory of munition j.

$$KILL_{klm} = \sum_i \sum_j \sum_n X_{ijklmn} \cdot YLD_{ijklmn} \qquad \text{for each } k, l, m \qquad \text{(A.7)}$$

The number of type k targets destroyed in distance band l during time period m must equal the number of missions flown against those targets times the effectiveness of the missions.

$KILL_{klm}$ = The number of type k targets killed (destroyed) during time period m in distance band l.

$$KILL_{klm} \leq TGT_{klm} \qquad \text{for each } k, l, m \qquad \text{(A.8)}$$

Limits the number of kills to the number of targets available.

TGT_{klm} = The number of targets available in distance band l during time period m. Available targets consist of targets that have never been killed plus those which have been killed during previous time periods and have regenerated (as a result of repair action or replacement).

$$TGT_{kl1} = STARTGT_{kl1} \qquad \text{for each } k,l \tag{A.9}$$

Sets the number of targets available for attack during time period one equal to the initial target set.

$STARTGT_{kl1}$ = initial target set.

$$TGT_{klm} = TGT_{kl(m-1)} - KILL_{kl(m-1)} + REGEN_{klm} \quad \text{for each } k,l,m > 1 \tag{A.10}$$

Sets the number of targets available for attack during time periods two and later equal to the number of targets surviving the previous time period plus those that regenerate during this time period. (Note: constraint 22 must be used for $REGEN_{klm}$ to be greater than zero.)

$REGEN_{klm}$ = the number of previously damaged/destroyed type k targets which become operational due to enemy repair or replacement activity during time period m. Targets regenerate in the same distance band they originate from.

$$\sum_i \sum_m (NAC_{im} \cdot ACCOST_i) + \sum_j (NMN_j \cdot MNCOST_j) + \sum_i \sum_m \sum_p (NSPR_{imp} \cdot SPRCOST_{ip}) \leq BUDGET \tag{A.11}$$

Limits the procurement of additional aircraft, spares, and munitions to a specified budget level.

$ACCOST_i$ = cost to procure an aircraft of type i.

$MNCOST_j$ = cost to procure a munition of type j.

$NSPR_{imp}$ = the number of spares resource type p necessary to support additional sorties for aircraft type i during time period m.

$SPRCOST_{ip}$ = the cost to procure spares resource p for each additional sortie flown by aircraft type i.

$$\sum_m ATTRIT_{im} \leq MAXATT_i \cdot \left[STARTAC_i + \sum_m FILL_{im} + \sum_m NAC_{im} \right] \quad \text{for each } i \tag{A.12}$$

Limits attrition for each aircraft type to a specified percent of the force. The percentage is typically calculated using Air Force planning factors and is input through the aircraft file. (See constraint 27 for an alternative maximum attrition constraint. If constraint 27 is used then 12 cannot be used.) This constraint can cause the problem to go infeasible when used in conjunction with constraints 13, 14, or 26. See section dealing with loss of feasibility due to constraint interactions.

$MAXATT_i$ = the maximum percentage of available type i aircraft which are allowed to be attrited during the war.

$$\sum_i [EAC_i + NAC_{im}] \geq MINSORT_m \cdot \left] \sum_i STARTAC_i + \sum_i \sum_m FILL_{im} + \sum_i \sum_m NAC_{im} - \sum_i \sum_{m-1} ATTRIT_{im} \right] \quad \text{for each } m \tag{A.13}$$

Forces the model to fly a minimum percentage of available sorties by time period. This constraint can cause the problem to go infeasible when used in conjunction with constraints 12, 14, 26 or 27. See section dealing with loss of feasibility due to constraint interactions.

$MINSORT_m$ = the required percentage of sorties which must be flown during time period m. This percentage applies to the sum of all aircraft sorties flown during time period m.

$$\sum_i KILL_{klm} \geq MKILL_{km} \qquad \text{for each } k, m \tag{A.14}$$

Forces the model to kill a user specified number of type k targets during each time period. (See constraint 26 for an alternative forced kill constraint.) This constraint can cause the problem to go infeasible when used in conjunction with constraints 12, 13, or 27. See section dealing with loss of feasibility due to constraint interactions.

$MKILL_{km}$ = the minimum number of targets of type k which must be killed during time period m .

$$NMN_j \leq UMN_j \qquad \text{for each } j \tag{A.15}$$

Production constraint, procure no more than some maximum number of munition j. This constraint is used to keep weapon procurement below production line maximum capacities.

UMN_j = *upper bound on munition j procurement.*

$$NMN_j \geq LMN_j \qquad \text{for each } j \tag{A.16}$$

Production constraint, procure at least some minimum number of munition j. This constraint is used to force buys of specific weapons to keep production lines open.

LMN_j = lower bound on munition j procurement.

$$\sum_m NAC_{im} \leq UAC_i \qquad \text{for each } i \tag{A.17}$$

Production constraint, procure no more than some maximum number of aircraft i.

UAC_i = upper bound on aircraft i procurement.

$$\sum_m NAC_{im} \geq LAC_i \qquad \text{for each } i \tag{A.18}$$

Production constraint, procure at least some minimum number of aircraft i.

LAC_i = lower bound on aircraft i procurement.

$$BAF = e_a \cdot \sum_i \sum_m NAC_{im} + e_m \cdot \sum_j NMN_j + e_s \cdot \sum_i \sum_m \sum_p NSPR_{imp} \qquad \text{(A.19)}$$

Calculate a budget awareness factor (BAF) using some small epsilon times the number of aircraft, spares, and munitions resources purchased. BAF causes the model to use existing aircraft and munitions prior to purchasing a new like item.

$e_a e_m e_s$ = small multipliers, typically in the 0.001 to 0.00001 range.

$$ATTRIT_{im} = \sum_j \sum_k \sum_l \sum_n X_{ijklmn} \cdot a_{ijklmn} \qquad \text{for each } i, m \qquad \text{(A.20)}$$

Calculate the experienced attrition for aircraft type i during time period m.

a_{ijklmn} = the expected attrition for a single mission using aircraft i, carrying munition j, against target k, in distance band l, during time period m, and weather state n.

$$NOTUSED \qquad \text{(A.21)}$$

$$REGEN_{klm} = \sum_{z=1}^{m-1} b_{kzm} \cdot KILL_{klz} \quad \text{for each } k, l, m > 1 \qquad \text{(A.22)}$$

b_{kzm} = the regeneration rate for each target type. The b's are calculated based on intel estimates of the enemy's repair capacity for that target type and the number of days in the time period. $b = .5$ means that half the type one targets destroyed during time period one will regenerate and be operational in time period two.

$$\sum_j \sum_k \sum_l \sum_n X_{ijklmn} = NSORT_{im} \qquad \text{for each } i, m \qquad \text{(A.23)}$$

Forces the model to fly a fixed number of sorties during each time period by aircraft type. This constraint may cause a loss of sensitivity to attrition. Used with constraint 24 to provide sortie distribution across the weather bands.

$NSORT_{im}$ = a fixed number of sorties to be flown by aircraft type i during time period m.

$$\sum_j \sum_k \sum_l X_{ijklmn} = PRCNT_n \cdot NSORT_{im} \qquad \text{for each } i, m, n \qquad \text{(A.24)}$$

Forces the model to distribute the fixed sorties across the weather bands. (See discussion with constraint number four.)

$$TVD = \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n X_{ijklmn} \cdot YLD_{ijklmn} \cdot TGTVAL_{klm} \qquad \text{(A.25)}$$

Forces the model to destroy a specified amount of target value. Developed for use with alternative objective functions to explore model sensitivities.

$$\sum_l \sum_m KILL_{klm} \geq MKILL_{km} \qquad \text{for each } k, m \qquad \text{(A.26)}$$

Forces the model to kill at least a specified number of type k targets by the end of time period m. The targets may be destroyed in any time period up to and including the current one. (See constraint 14 for an alternative forced kill constraint.) This constraint can cause infeasibility when used in conjunction with constraints 12, 13, or 27. See section dealing with loss of feasibility due to constraint interactions.

$$\sum_m ATTRIT_{im} \leq MAXATT_i \qquad \text{for each } i \qquad \text{(A.27)}$$

Limits attrition for each aircraft type to a specified number of aircraft. (Constraint number 12 limits attrition to a specified percentage of the force available.) This constraint can cause infeasibility when used in conjunction with constraints 13, 14, or 26. See section dealing with loss of feasibility due to constraint interactions.

$$\sum_l \sum_n X_{ijklmn} = AGSORT_{im} \qquad \text{for each } i, m, j = 58, k = 86 \qquad \text{(A.28)}$$

Forces the model to fly a specified number of sorties against a generic ground based threat oriented target ($k = 86$). The sorties are specified by aircraft type and time period. Munition 56 is a dummy munition load with no cost.

$AGSORT_{im}$ = input number of sorties flown by aircraft type i against ground threat targets during time period m.

$$\sum_{l}\sum_{n} X_{ijklmn} = AASORT_{im} \quad \text{for each } i, m, j = 58, k = 86 \qquad \text{(A.29)}$$

Force the model to fly a specified number of sorties against a generic airborne threat oriented target ($k = 87$). The sorties are specified by aircraft type and time period. Munition 56 is a dummy munition load with no cost.

$AASORT_{im}$ = Input number of sorties flown by aircraft type i against airborne threat targets during time period m.

$$\sum_{j}\sum_{k}\sum_{l}\sum_{n} X_{ijklmn} \leq \sum_{m}(INSPR_{imp} + NSPR_{imp}) - \sum_{j}\sum_{k}\sum_{l}\sum_{m-1}\sum_{n} X_{ijklmn} \quad \text{for each } i, m, p \qquad \text{(A.30)}$$

Limits sorties during each time period to those supportable with spares.

$INVSPR_{imp}$ = inventory of type p spares resource available for use on aircraft type i during time period m. Inventory unused during time period m roles over into $m + 1$. This inventory represents the spares supported sorties available at no additional cost.

$NSPR_{imp}$ = new type p spares resource procured during time period m for use on aircraft type i.

$$\sum_{m} NSPR_{imp} \leq UPPSPR_{ip} \quad \text{for each } i, p \qquad \text{(A.31)}$$

Upper bound. Limits the procurement of additional spares resources based on production constraints or availability.

$UPPSPR_{ip}$ = upper bound on total new type p spares resource procurable for use on aircraft type i.

$$\sum_m NSPR_{imp} \geq LOWSPR_{ip} \quad \text{for each } i,p \tag{A.32}$$

Lower bound. Forces procurement of at least some minimum number of spares resources. Typically set to zero to prevent selling spares back to procure additional munitions or aircraft.

$LOWSPR_{imp}$ = lower bound on total new type p spares resources procurable for use on aircraft type i.

A.2.4 Constraint Interactions The current formulation of the problem contains several constraints which are used to force the solution to attain some operational goals while staying within budgetary limits. In most cases these goals can be fully attained when taken one at a time. However, the problem being solved often requires attainment of several of these goals simultaneously which can lead to unresolvable conflicts or, in LP terminology, an infeasible solution. This section will briefly explore the operational goals which are modeled, their associated constraints, their interactions and methods of reducing conflicts between goals.

First let's look at a notional set of operational goals which can be modeled using TAM. To attain some operational objective (achieve air superiority, win the war) we need to satisfy the following goals: 1) We must fly at least 90 percent of the available sorties, 2) suffer no more than 30 percent attrition over the duration of the war, and 3) destroy at least 80 percent of the assigned targets. These goals are specified in TAM as constraints which must be satisfied. For example, constraint 13 would require that at least nine tenths of the available force structure be used during each time period ($MINSORT_m = 0.9$). This equates to flying 90 percent of the available sorties. Similarly constraint's 12 and 27 restrict the number of each type aircraft that can be lost to attrition during the war (max attrition constraint), and constraint 's 14 and 26 force at least a minimum number of target kills by target type and time period (min kill constraint).

From an operational perspective it is clear that these goals interact with each other, i.e. as you fly additional sorties, you can destroy more targets but you also suffer attrition consistent with the target defenses. The operational goals dictate certain levels of target destruction to attain the objective of winning the war. The time period necessary to achieve this objective is frequently dictated as well, and is often set to the initial week or two of the war. Unfortunately, resource limitations, aircraft, munitions, spares, etc. often do not allow for attaining these objectives and conflicts between the constraints arise. It is then necessary to determine the level of destruction which is possible given the resources at hand. In TAM this equates to reducing the min kill constraint to an achievable level. Once this level of destruction is identified, it is then possible to determine if increasing the allowable level of attrition will significantly improve the solution. It is also at this point in the analysis where increasing the budget available provides insight into the marginal benefit of buying additional aircraft, munitions, or spares resources.

Care should be taken when analyzing the results of the model to ensure that the problem is not unconstrained (too many resources for the job). If all of the targets are destroyed, or part of the budget dollars available are not spent, or aircraft are not fully utilized, then the results of the analysis need to be carefully looked at to ensure there are no flaws in the way resources are being utilized. For example if the budget provides the capability to destroy all the targets, then it is possible that aircraft and munition loads are being used incorrectly (less effective missions) just to fly the sorties required by the minimum sortie constraint. Similar conditions can exist which will cause higher attrition missions to be selected due to an excess of aircraft.

Appendix B. *Suggested Reformulation of TAM*

B.1 PARAMETERS :

$i - a$ = aircraft 1 to 20

$j - m$ = munitions 1 to 56

$k - k$ = targets 1 to 86

$l - d$ = distance bands 1 to 4

$m - t$ = time bands 1 to 4

$n - w$ = weather bands 1 to 6

$p - s$ = spares 1 or 2 for each aircraft

B.2 Decision Variables:

Generally everything is now reduced in terms of the decision variables:

-(X) sorties with aircraft a, carrying munition m, to target k, in distance band d, during time band t, in weather w.

-NAC new aircraft purchased

-NMN new munitions purchased

-NSPR new spares purchased

B.3 Objective Function:

MAximize TVD (Target Value Destroyed) =

$$\sum_{a}\sum_{m}\sum_{k}\sum_{d}\sum_{t}\sum_{w} X_{amkdtw} \cdot YLD_{amkdtw} \cdot TGTVAL_{kdt} \; - \; (e_a \cdot \sum_{a}\sum_{t} NAC_{at}) - (e_m \cdot \sum_{m} NMN_m) - (e_s \cdot \sum_{a}\sum_{t}\sum_{s} NSPR) \qquad (B.0)$$

This objective function combines the original objective (1) with constraint 20. As mentioned in chapter 4 consider removing the budget awareness factors from the model.

B.4 Constraints:

Represent blocks of constraints that are selected by the analyst depending on the analysis to be done.

1. Total aircraft constraints per period. This block of constraints combines the original constraints 2,3,4, and 21.

$$\sum_m \sum_k \sum_d \sum_w \frac{X_{amkdtw}}{TS_{amkdtw}} \leq STARTAC_a + \sum_t FILL_{at} + \sum_{t-1} NAC_{at} - \sum_{t-1} \left(\sum_m \sum_k \sum_d \sum_w X_{amkdtw} \cdot a_{amkdtw}\right) \quad \text{for each } a,t \qquad \text{(B.1)}$$

2. Optional constraint to force sorties to fly across all weather bands. This constraint combines old constraints 5 and 21.

$$\sum_m \sum_k \sum_d \frac{X_{amkdtw}}{TS_{amkdtw}} \leq PRCNT_w \cdot \left[STARTAC_a + \sum_t FILL_{at} + \sum_t NAC_{at} - \sum_{t-1} (X_{amkdtw} \cdot a_{amkdtw})\right] \text{for each } a,t,w \qquad \text{(B.2)}$$

3. Total munitions used across the war must be less than starting inventory plus those newly purchased. This constraint combines old constraints 6 and 7.

$$\sum_a \sum_k \sum_d \sum_w X_{amkdtw} \cdot LOAD_{amd} \leq STARTMN_m + NMN_m \quad \text{for each } m \qquad \text{(B.3)}$$

4. Total targets constrained per distance and time band. These constraints combine old constraints 8,9,10,11 and 23.

$$\sum_a \sum_m \sum_w X_{amkdtw} \cdot YLD_{amkdtw} \leq STARTGT_{kdl} \quad \text{for each } k,d,t = 1 \qquad \text{(B.4)}$$

$$\sum_{a}\sum_{m}\sum_{w} X_{amkdtw} \cdot YLD_{amkdtw} \leq STARTGT_{kd(t=1)} - KILL_{kd(t-1)} \sum_{z=1}^{t-1} b_{kzt} \cdot KILL_{kdz} \quad \text{for each } k,d,t \qquad \text{(B.5)}$$

5. Budget constraint. Remains unchanged from the original formulation constraint 12 except for the subscript changes.

6. Maximum attrition rates per aircraft. This constraint combines old constraint blocks 13 and 28. Either old constraint 12 (unchanged) is selected or this constraint is used to limit attrition.

$$\sum_{a}\sum_{t}\sum_{d}\sum_{w}(X_{amkdtw} \cdot a_{amkdtw}) \leq MAXATT_a \cdot (STARTAC_a + \sum_{t} FILL_{at} + \sum_{t} NAC_{at}) \text{ for each } a \qquad \text{(B.6)}$$

7. Minimum sorties per time period that must be flown. This constraint replaces old constraint 14.

$$\sum_{a}(EAC_{at} + NAC_{at}) \geq MINCOST_t \cdot \left[STARTAC_a + \sum_{a}\sum_{t} FILL_{at} + \sum_{a}\sum_{t-1}(X_{amkdtw} \cdot a_{amkdtw})\right] \text{ for each } t \qquad \text{(B.7)}$$

8. Minimum targets killed per time period. This constraint is a choice between old constraints 15 and 27.

9. Minimum aircraft sorties per weather band. This constraint combines old constraints 24 and 25. It is used in place of constraint 3 to force a fixed number of sortie types to fly across a wather band.

$$\sum_{m}\sum_{t}\sum_{d} X_{amkdtw} = PRCNT_w \cdot (NSORT_{at}) \quad \text{for each } a,t,w \qquad \text{(B.8)}$$

10. Specifies Target Value Destroyed, this is used with alternate objective functions to explore the sensitivity of other items of interest. This constraint duplicates old constraint 26.

11. Fixes a specific number of air-to-ground sorties. This constraint duplicates old constraint 29.

12. Fixes a specific number of air-to-air sorties. This constraint duplicates old constraint 30.

13. Limits the sorties flown during a time period to those supported by spares. This constraint duplicates old constraint 31.

14. Upper and lower bounds on procurement of aircraft, munitions, and spares. These constraints replace old constraints (16,17), (18,19), and (32,33). These constraints can be handled through the bounds section or through the ranges section and need not be explicitly stated.

$$LMN_m \leq NMN_m \leq UMN_m \quad \text{for each } m \tag{B.9}$$

$$LAC_a \leq NAC_a \leq UAC_a \quad \text{for each } m \tag{B.10}$$

$$LSPR_{as} \leq NSPR_{az} \leq USPR_{as} \quad \text{for each } a, s \tag{B.11}$$

Bibliography

1. Adler I., and others. *An Implementation of Karmarkar's Algorithm for Linear Programming.* Technical Report 86-8, Operations Research Center, University of California, Berkeley, CA, 1986.
2. Beale, E.P.L. *Mathematical Programming in Practice.* Great Britain: Pittman & Sons LTD., 1968.
3. Bazaraa, Mohktar S. and John J. Jarvis. *Linear Programming and Network Flows.* New York: John Wiley and Sons, 1977.
4. Benichou, M. and others, "The Efficient Solution of Large-Scale Linear Programming Problems–Some Algorithmic Techniques and Computational Results," *Mathematical Programming 13: 280-322* (December 1977).
5. Bracken, Jerome and others. *Report of Combat Consumption Modeling Improvement Panel.* July 1980. Contract No. MDA903-79-C0320. Institute for Defense Analyses, Program Analysis Division, Arlington VA. 22202.
6. Bradley, Gordon H. Gerald G. Brown, and Glenn W. Graves, "Redundancy in Mathematical Programming," *Lecture Notes in Economics and Mathematical Systems.* Volume 206, edited by Mark H. Karwan and others. New York: 1983.
7. Brearley, A., G. Mitra, and H. Williams, "Analysis of Mathematical Programming Problems Prior to Applying the Simplex Algorithm," *Mathematical Programming 8:* 54-83 (1975).
8. Brown, Gerald G. and Glen W. Graves, *Elastic Programming: A New Approach to Large-Scale Mixed-Integer Optimization*, ORSA/TIMS, Las Vegas (1975).
9. Brown, Gerald G. and David S. Thomen, "Automatic Identification of Generalized Upper Bounds in Large-Scale Optimization Models," *Management Science 26:* 1166-1184 (November 1980).
10. Brown, Gerald G. and William G. Wright, "Automatic Identification of Embedded Network Rows in Large-Scale Optimization Models," *Mathematical Programming 29:* 41-46 (May 1983).
11. Brown, Gerald G., Richard D. McBride, and R. Kevin Wood, "Extracting Embedded Generalized Networks From Linear Programming Problems," *Mathematical Programming 32:* 11-31 (May 1985).
12. Charnes, A. and others. *Complexity and Computability of Solutions to Linear Programming Systems.* February 1980. ONR Contract N00014-75-C-0569. Center for Cybernetic Studies, The University of Texas at Austin, Austin TX, 1980.
13. Chief of Staff of the Royal Air Force. *RAF Conventional Weapons Procurement Model.* CS(PRN) Memo #406. London: March 1980.
14. Chvatal, Vasek.*Linear Programming.* San Francisco: W.H. Freeman and Co., 1983.
15. Cooper, Leon and David I. Steinberg. *Methods and Applications of Linear Programming.* Philadelphia PA: W.B. Saunders and Co. 1974.

16. Dantzig, George B. *Linear Programming and Extensions.* Princeton NJ: Princeton University Press, 1963.

17. Dantzig, George B. and William Orchard-Hays. "The Product Form of the Inverse in the Simplex Method," *Mathematical Tables and Other Aids to Computation 8:* 64-67 (1955).

18. Dantzig, George B. and R. M. Van Slyke. "Generalized Upper Bounding Techniques,"*Journal of Computer and System Sciences 1:* 213-226 (1967).

19. Dantzig, George B. and Philip Wolfe, "Decomposition Principle for Linear Programs," *Operations Research 8:* 101-111 (January 1960).

20. Forrest, J.J.H. and J.A. Tomlin, "Updating Triangular Factors of the Basis to Maintain Sparsity in the Product Form Simplex Method," *Mathematical Programming 2:* 263-278 (1972).

21. Garey, Michael R. and David S. Johnson. *Computers and Intractibility, A Guide to the Theory of NP-Completeness.* San Francisco: W.H. Freeman and Co. 1979.

22. Graves, G.W. and R.D. McBride, "The Factorization Approach to Large-Scale Linear Programming," *Mathematical Programming 10:* 91-110 (February 1976).

23. Grotte, Jeffrey H. and Paul F. McCoy. *A Model for the Analysis of Stockpile Production Base Tradeoffs.* March 1979. Contract No. MDA903-79-C0018. Institute for Defense Analyses, Program Analysis Division, Arlington, VA. 22202.

24. Harris, Paula M.J. "Pivot Selection Methods of the Devex Code," *Mathematical Programming Study 4:* 30-57 (1975).

25. Lasdon, Leon S. *Optimization Theory for Large Systems.* New York: The Macmillan Co., 1970.

26. Karmarkar, N. "A New Polynomial Time Algorithm for Linear Programming," *Combinatorica 4:* 373-395, (1984).

27. Ketron Management Science Inc. *MPS III Mathematical Programming System-Dealer Manual.* Ketron Inc., Arlington VA, August 1988.

28. Khachian, L.G. "A Polynomial Algorithm in Linear Programming," *Soviet Math. Doklady,* 20 (1979).

29. Klee, V. and G.J. Minty. "How Good is the Simplex Algorithm," *Inequalities-III* O. Shisha ed. New York: Academic Press 159-175 (1972).

30. Lord, P. *An Examination of the United States Air Force Optimal Nonnuclaer Munition Procurement Model,* MS Thesis in Operations Research, Naval Postgraduate School, Monterey (1982).

31. Might, Robert J. "Weapon System Acquisition and Theater Level Analysis," to be published in *Management Science.*

32. Might, Robert J. "Decision Support for Aircraft and Munitions Procurement," *Interfaces 17:* 55-63 (October 1987).

33. McBride, Richard D. *Factorization in Large-Scale Linear Programming,* PHD Dissertation University of California, Los Angeles (1973).

34. Murtaugh, Bruce A. and Michael A. Saunders. *MINOS 5.0 User's Guide*, December 1983. Technical Report SOL 83-20. Systems Operations Laboratory, Department of Operations Research, Stanfod University, Stanford CA.

35. Murty, Katta G. "Computational Complexity of Parametric Linear Programming," *Mathematical Programming 19:* 213-219 (September 1980).

36. Nazareth, J.L. *Computer Solutions of Linear Programs.* New York: Oxford University Press, 1987.

37. Nelson, Carroll, President KETRON Inc. Phone Interview and Personal Correspondence 16-19 August, 1988.

38. Olson, CMDR Michael Untitled PhD dissertation Naval Postgraduate School, Monterey CA. Due June 1989.

39. Orchard-Hays, William. *Advanced Linear Programming Computing Techniques.* New York: McGraw Hill, 1968.

40. Reid, J.K. "A Sparsity-Exploiting Variant of the Bartels-Golub Decomposition for Linear Programming Bases," *Mathematical Programming 24:* 55-69 (1974).

41. Reybrock, Nick, Major/USAF/AFCSA/SAGM Personal Interview at AFCSA 5-9 May, 1988.

42. Reybrock, Nick, Major/USAF/AFCSA/SAGM Phone Interview 11 December 1988.

43. Schnabel, Robert B. "Parallel Computing In Optimization," *Computational Mathematical Programming*, edited by Klaus Schittkowski. Bad Windsheim FRG: Springer-Verlag Berlin Heidelberg, 1985.

44. Strang, Gilbert. *Linear Algebra and its Applications.* Orlando FL: Harcourt Brace Jovanovich Publishers, 1988.

45. Tomlin, J.A. "On Scaling Linear Programming Problems," *Mathematical Programming Study 4:* 146-166 (1975).

46. Winston, Wayne L. *Operations Research: Applications and Algorithms.* Boston: Duxbury Press, 1987.

47. Wirths, K. *A Thesis*, MS Thesis in Operations Research, Naval Postgraduate School, Monterey (1988).

48. Wolfe, Philip "A Technique for Resolving Degeneracy In Linear Programming," *Journal of the Society for Industrial Application of Mathematics 2:* Volume II. 205-211 (June 1963).

Vita

Major Jack A. Jackson Jr. [redacted] He spent one year at the University of Nebraska prior to entering the USAF Academy in 1972. Major Jackson graduated from the Academy and was commisioned as a Second Lieutenant on 2 June 1976.

After graduation from Undergraduate Pilot Training at Reese AFB Texas, Major Jackson was assigned as an Instructor Pilot in the T-38 aircraft to the 64FTW and then to the 12FTW as a PIT Instructor Pilot at Randolph AFB. He has over 1700 hours in the T-38 aircraft.

In 1983 Major Jackson was assigned to the 48TFW, at RAF Lakenheath in the United Kingdom. During this tour he was a Flight Commander and Instructor Pilot in the 492TFS; earned the Meritorious Service Medal and Air Medal; accumulated over 1000 hours in the F-111F; and was promoted to his current rank. He entered the School of Engineering, Air Force Institute of Technology at Wright-Patterson AFB Ohio in 1987, and he is currently assigned to the Air Force Center for Studies and Analysis at the Pentagon.

[redacted]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA205847

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GST/ENS/89M-7	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology (AU) Wright-Patterson AFB, Ohio 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFCSA 8b. OFFICE SYMBOL (If applicable) SAGM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Pentagon,ADM; Virginia 22015	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
A TAXONOMY OF ADVANCED LINEAR PROGRAMMING TECHNIQUES AND THE THEATER ATTACK MODEL

12. PERSONAL AUTHOR(S)
JACK A. JACKSON Jr. MAJOR, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS THESIS	FROM _____ TO _____	MARCH 1989	92

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	LINEAR PROGRAMMING SIMPLEX METHOD MATHEMATICAL MODELS
12	04		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

MAJOR JOSEPH R. LITKO, PhD

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL MAJOR JOSEPH R. LITKO, Assistant Prof	22b. TELEPHONE (Include Area Code) 513-255-3362	22c. OFFICE SYMBOL AFIT/ENS

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

BLOCK 19 CONTINUED:

AFIT/GST/ENS/89-7

Abstract

The Theater Attack Model (TAM) is a large-scale linear program (LP) used to aid senior decisionmakers in making the tough budget and procurement decisions for the United States Air Force. TAM, as currently configured, can generate matrices with 9 million variables. The CPU-time to run this model is enormous. Advanced LP techniques are examined to reduce TAM's CPU-time.

A taxonomy of advanced LP techniques results from a review of major characteristics of these techniques. Promising opportunities to reduce TAM's solution time are tested and compared to a standard simplex benchmark. Karmarkar's algorithm on the KORBX (AT&T) system is tested and compared with simplex-based techniques through an absolute comparison of solution time and a relative comparison based on machine capabilities.

Recommendations for reducing TAM's CPU-time are outlined with the hope of saving the government money in computer time.