AN IMPROVED SOLUTION METHODOLOGY FOR THE ARSENAL EXCHANGE MODEL (AEM)

I. Introduction

Background

The Arsenal Exchange Model (AEM) is an aggregated, two-sided, strategic exchange model used primarily for strategic weapon system analysis, strategic nuclear policy and arms control analysis, and general strategic calculations (2:4). It performs allocations of weapons to targets in an attempt to maximize the amount of damage done to targets. Since both the number of weapons and targets are integer values, the AEM formulates the problem as an integer programming (IP) problem. The variables of the problem represent weapon/target combinations that can achieve a desired level of damage expectancy. Often, more than a million variables are considered, and finding an optimal integer solution would require the use of too much time and resource.

Fortunately, an optimal integer solution is not required by the users. The users prefer a good feasible integer solution quickly. Currently this is accomplished by solving the linear programming (LP) relaxation of the IP and using a heuristic to find a feasible integer solution (6). There are some problems, however, with the current LP solver used by the AEM.

In creating the LP, the AEM generates weapon constraints, target constraints, and goal or hedge constraints based on user specified inputs. The AEM solver currently uses a combination of column generation, generalized upper bounding (GUB), decomposition, Gauss-Jordan elimination, and compact tableau storage. Like all column generating techniques, the AEM solves a sequence of LPs. The AEM solves the original LP by first solving an LP with a subset of the strategies. When that subset has been solved to optimality, the AEM's column generator uses the basis inverse matrix of that solution to generate strategies that will improve the current objective function value. These columns are then incorporated into a new LP with the basis matrix, and the AEM solves this new LP to optimality. The AEM continues this iterative process until all possible columns have been generated, or until no new columns will increase the value of the current objective function. For large, complex AEM applications, precision and speed problems are noticeable (5:881). The AEM solver uses several techniques to alleviate the precision problems. Among these are re-inversion and backward pivoting. Although these techniques alleviate a large portion of the precision problems, those caused by redundant or identical constraints are not entirely corrected.

The hedge constraints are used to ensure the solution meets certain force employment policies (2:30). Thus, hedge constraints are used to incorporate goals into the model. At certain points during the solution procedure, the hedge constraints can be identical to other constraints already generated by the AEM. These identical or redundant constraints cause singularity problems in the Gauss-Jordan elimination algorithm (see Chapter II). Since singular matrices cannot be inverted, the AEM multiplies the right hand side of the hedge constraints by 0.9995 (3). The new matrix, while nearly singular, can now be inverted. While the adjustment factor does eliminate the singularity problem, it adds to the precision problem in two ways. First, it changes the problem being solved. The LP will now solve a problem that ensures only 99.95% of force employment policies are met. Second, the near singularity of the new matrix causes mathematical precision problems during the Gauss-Jordan pivots. In addition, the Gauss-Jordan inverter algorithm also has speed problems.

The Gauss-Jordan inverter algorithm is used because of its robustness. Although the Gauss-Jordan routine can invert the matrix, it is not as efficient as other routines (9:25). That is, the robustness of a Gauss-Jordan inverter is at the expense of efficiency. Modern, state-of-the-art routines recognize the near singularity of the matrix and will not invert it (7:31). If a technique can be found to eliminate the identical constraints, then the correction factor can be eliminated and a more precise and efficient inversion routine can be used.

Model Description

This section gives a brief description of the AEM. It describes the user inputs to the model, and how the mathematical model is formulated.

Inputs. There are two basic types of user inputs for the AEM. The first type consists of the resources available to each side. These include the weapons to be used and the targets to be attacked. The second type, a set of user controls known as hedges, set goals and requirements for the allocation of the resources.

Weapons and Targets. The user is allowed to define up to 100 weapon types (2:39). Some examples of weapon types are B-52s, Minuteman IIIs, and Trident D-5s. The user inputs the number of weapons of each type, as well as a number of other weapon characteristics, such as, the number of warheads delivered by each weapon, circular error probable (CEP), reliability, and availability. Similarly, the target set is broken up into classes. The limit on the number of target classes is 2500 minus the number of weapon types (2:39). The AEM assumes each weapon type is seen by the other side as a target class. Within each target class, the user sets the number of targets in that class and inputs a variety of target characteristics including hardness, size, and value.

Hedges. Hedge constraints are input by the user to ensure the AEM meets certain force employment policies. The user also determines how the AEM will use these inputs. The input can be used as a goal to be attained or as a requirement that must be met. In the first, the user does not associate a priority with the goal, and the AEM tries to minimize all of the deviations from the user's goals through goal programming techniques (see Chapter II). Failure to meet a goal affects the goodness of the allocation. In the second, the user specifies a priority with the requirement, and the AEM attempts to satisfy the user specified requirements in priority order using pre-emptive goal programming techniques (see Chapter II). In this case, failure to meet a requirement affects the feasibility of the allocation.

Mathematical Model Formulation. The mathematical model formulated by the AEM is a linear program (LP), which is shown in Equations (1-1) to (1-4). In the LP, the objective function (1-1) maximizes damage expectancy subject to three sets of constraints. The first set of constraints (1-2) ensures that the number of weapons used from each weapon class does not exceed the number of weapons available in that class. The second set of constraints (1-3) ensures that the total number of targets attacked in each target class does not exceed the total number of targets in that class. Finally, the third set of constraints (1-4) specifies user input hedges. The hedge constraints are added to the LP using goal programming (see Chapter II). After formulating the model, the AEM uses a column generating algorithm to solve the LP.

Each column in the LP is a strategy. A strategy is a weapon/target combination that achieves a certain damage expectancy. For example if two warheads of weapon type 1 can be used against one target in target class A with a damage expectancy of 0.73, the strategy column in the LP would look like:

$$\begin{array}{cl}
\hline
0.73 * \mathrm{STRAT}_j & \text{Objective function} \\
\hline
2 * \mathrm{STRAT}_j & \text{Weapon constraint 1} \\
0 & \\
\vdots & \\
0 & \\
\hdashline
1 * \mathrm{STRAT}_j & \text{Target constraint A} \\
0 & \\
\vdots & \\
0 & \\
\hline
\end{array}$$

where STRAT_j is the variable representing the number of times weapon/target combination j is used, and $\mathrm{DE}_j = 0.73$, $\mathrm{W}_{j1} = 2$, and $\mathrm{T}_{jA} = 1$. The entire LP formulation then has the following mathematical representation:

$$\text{Max} \qquad \sum_j \mathrm{STRAT}_j * \mathrm{DE}_j - \sum_g \mathrm{M}_g * \mathrm{DIFF}_g \tag{1-1}$$

$$\text{Subject To} \quad \sum_j \mathrm{W}_{jl} * \mathrm{STRAT}_j \le \mathrm{WEP}_l \qquad \text{for } l = 1,\dots,\mathrm{o} \tag{1-2}$$

$$\sum_j \mathrm{STRAT}_j * \mathrm{T}_{jk} \le \mathrm{TAR}_k \qquad \text{for } k = 1,\dots,\mathrm{p} \tag{1-3}$$

$$\sum_j \mathrm{Hedge}_g(\dots) + \mathrm{DIFF}_g = \mathrm{HEG}_{0g} \qquad \text{for } g = 1,\dots,\mathrm{q} \tag{1-4}$$

$$\mathrm{STRAT}_j \ge 0 \quad \text{for } j = 1,\dots,\mathrm{n}$$

Where:

m	= the total number of constraints = o + p + q
n	= the number of allowable strategies
o	= the number of weapon types
p	= the number of target classes
q	= the number of hedges
STRAT_j	= the number of times strategy j is chosen

DE_j	= the damage expectancy of strategy j
M_g	= the penalty amount for not achieving hedge g
$DIFF_g$	= the amount by which hedge g was not achieved
W_{jl}	= the number of type l weapons used in strategy j
WEP_l	= the total number of type l weapons
T_{jk}	= the number of type k targets attacked in strategy j
TAR_k	= the total number of type k targets
HEG_{og}	= the level to be achieved by hedge g
$Hedge_g(\ldots)$	= a linear function of the previously defined variables

Problem Statement

The AEM suffers from numerical and precision problems that increase solution time and degrade the possibility to meet policy specified goals (5:881).

Purpose Statement

The purpose of this research is to improve the LP solution methodology in the AEM. This new methodology should be computationally faster and contain less precision error than the current methodology.

Scope

The two most important factors associated with running the AEM are solution time and precision. To determine if the improved methodology is better than the current one, these two parameters are measured. Solution time is defined as the time required to solve to optimality all of the LPs generated by the AEM. Time is measured in CPU seconds. Precision is measured by comparing the damage expectancy and goal performance of the continuous solutions found by each methodology.

Format

Chapter II provides an explanation of goal programming in general and its implementation in the AEM. Chapter II also presents an explanation of solution techniques that have been used to overcome problems similar to those encountered by the AEM. The improved solution methodology is presented in Chapter III. Chapter IV discusses implementation and test results. Chapter V presents conclusions and recommendations.