[1]

WORKING DRAFT
TOWARDS THE REAL TIME SOLUTION OF STRIKE FORCE ASSET ALLOCATION PROBLEMS

VINCENT C. LI
C0L0558@YAHOO.COM , GUY L. CURRY
G-CURRY@TAMU.EDU , AND E. ANDREW BOYD
ABOYD@PROSRM.COM

Abstract

The strike force asset allocation problem consists of grouping strike force assets into packages and assigning these packages to targets and defensive assets in a way that maximizes the strike force potential. Integer programming formulations are developed, semi-real and random problems are built up, and computational results via the CPLEX MIP Solver are presented.

[1]Work Sponsored in part by the Office of Naval Research grant numbers 00014-96-1-0316 and 00014-96-1-0912.

1. Introduction. Strike force planning is one of the fundamental problems faced by the armed services. In a strike force planning problem, there are a set of potential targets and a collection of assets that may be allocated to targets. There may also be a set of threats which serve as the defenders of the targets. The goal is to determine the allocation and routing of strike force assets so as to maximize strike effectiveness while limiting potential damage to the strike force from the defenders.

The strike force planning problem is solved before the engagement, although the problem of strike force management during the engagement is also a question of practical importance. Current practice requires that the strike force planning problem be solved on a time scale of a few hours to one-half a day. With recent advances in information processing technology, especially in parallel computation, the potential for real-time strike force planning – on the order of a few minutes – is now realizable.

The strike force planning problem includes routing and allocating the available assets. The routing problem has received attention in several papers (Boroujerdi et al. 1993; Boroujerdi 1994; Zuniga et al. 1994; Lee 1995; Zuniga and Gorman 1995). Zuniga and Gorman (1995) discuss strike route optimization and threat site overflight modeling. They present some general graph theoretic techniques for route optimization. Zuniga et al. (1994) focus on the inter-dependent joint routing problem. They analyze routing a collection of assets to a collection of targets. Lee (1995) constructs a network and uses a shortest path algorithm to obtain a very good route for the strike force aircraft.

As to the asset allocation problem, Matlin (1970), Eckler and Burr (1972) and Cheong (1985) give reviews on the missile allocation problem. Cho (2000) discusses how to allocate a given number of weapons to targets with target-dependent kill probabilities. While most of the literature models the weapon-target allocation problem in a stationary way, Hosein (1989) treats it as a dynamic-nonlinear resource-allocation problem. Hosein and Athans (1990) give analytic results for the dynamic nonlinear weapon-target allocation problem in simple cases. They give asymptotic results when the number of targets becomes very large.

Work by Jakobovits (1995) highlights the core of the asset allocation problem. Jakobovits separates a strike plan into an *attack plan*, which focuses on primary targets, and a *suppression plan*, which seeks to suppress defensive threats. He addresses the asset allocation problem in this general setting and solves some small problem instances, but he points out that larger instances quickly become very difficult to solve.

In this paper, we omit routing information, and consider two models of the striking asset allocation problem depending on whether or not suppression of defensive threats is involved. In the first model, we formulate the problem as a large-scale integer program consisting of a collection of 0-1 knapsack constraints superimposed on a collection of disjoint Type 1 special ordered sets (SOS). A knapsack constraint has the form $ax \leq b$, where x is a column vector of 0/1 variables, a is a nonnegative row vector and $b \geq 0$, and special ordered sets are sets of non-negative variables that are required to sum to 1. A Type 1 SOS is when each variable is binary so the constraint is one of multiple choice (Greenberg, 2001) . The second model further incorporates precedence constraints in addition to a collection of knapsack constraints and SOS constraints.

As the computational results demonstrate, the problem structures afforded by these constraints yield integer programs that can be solved close to 1% from the best bound of the optimal solutions very quickly via the Parallel MIP Solver (MIP is an abbreviation for mixed integer programming) of CPLEX 6.6 (1999), even for relatively large problem (15,000 binary variables and 90 constraints) instances. In addition to the models presented and studied computationally, a set of semi-real problems has been developed based on data procured from the United States Geological Survey. In addition, a set of randomly generated problems were created. It is hoped that other researchers will find this set of test problems interesting and useful for future research in this problem area. The problems and contour maps are available upon request.

The paper is organized as follows. In Section 2, the asset allocation problem formulations are described. Section 3 discusses generating semi-real and random problems. Section 4 presents computational results for both models. Conclusions and future research directions are presented in Section 5.

2. Formulations of Problems. A practical aspect of strike force asset allocation is that often more than one asset is required to attack a target or suppress a threat. We will use threats and defenders interchangeably in this paper. There are several types of strike force assets represented in the form of different aircraft and missiles. To illustrate, a radar site might require two missiles of type I, and a naval base might require one aircraft of type I and one missile of type II. Moreover, there might be other combinations which can be used to attack a target or suppress a threat. In this paper, the collection of assets capable of destroying a target and a threat will be referred to as an *attack package* and a *suppression package*, respectively. Model 1 considers only targets. This model is generalized to incorporate threat suppression as Model 2.

2.1. Strike Force Asset Allocation Problem without Suppression. Let C be the set of all asset classes available and T be the set of all targets available. Identical assets are referred to as the same *asset type* (independent of location). An *asset class* is an asset type with a specific location. For example, two identical aircraft in different locations are of the same asset type but different asset classes. The cost of assigning the same asset at different locations to a given target may be different due to the distance between the target and the asset. For simplicity, assume that an asset close to its corresponding target will have less opportunity for asset loss, therefore, the distance between each asset in an attack package and the corresponding target contributes location dependent values associated with the attack package value. Before stating the formulation, define some notation as follows:

A : the set of all possible target attack packages,

A_t : a subset of A which consists of all attack packages associated with target t where the cardinality of A_t is $|A_t|$,

g_{ta} : the profit associated with assigning attack package a to target t, where $a \in A_t$,

e_{cta} : number of class c assets contributing to attack package a for target t, where $a \in A_t$,

b : a column vector of size $|C|$, where b_c is the number of assets available in asset class c,

x_{ta} : equals 1 if attack package a is assigned to target t, and equals 0 otherwise, where $a \in A_t$.

The goal is to maximize the damage yield which is both target and package dependent by the assignment of attack packages subject to asset class configurations and availability constraints. Each attack package has an associated binary variable in the model, and each attack package affects the objective value in an additive manner according to its coefficient if the attack package is chosen; each attack package contributes nothing to the objective value if it is not chosen. It is assumed each target is assigned at most one attack package. This model can be described as follows.

Model 1

$$\max \sum_{t \in T} \sum_{a \in A_t} g_{ta} x_{ta} \tag{2.1}$$

$$s.t. \sum_{t \in T} \sum_{a \in A_t} e_{cta} x_{ta} \leq b_c \quad \forall c \in C, \tag{2.2}$$

$$\sum_{a \in A_t} x_{ta} \leq 1 \quad \forall t \in T, \tag{2.3}$$

$$x_{ta} \in \{0, 1\} \quad \forall t \in T,\ a \in A_t. \tag{2.4}$$

The objective function (2.1) assigns a profit g_{ta} for package a corresponding to target t ($a \in A_t$) if the package is chosen ($x_{ta} = 1$). Constraints (2.2) guarantee that whatever attack packages are chosen, the overall demand for each asset class will not exceed the supply. Constraints (2.3) guarantee that no target will be assigned more than one attack package which is an assumption for Model 1.

2.2. Strike Force Asset Allocation Problem with Suppression. Next, threat suppression is incorporated into Model 1. Each threat can defend one or more targets in its neighborhood. Each target can have one or more threats protecting it as well. To effectively destroy the target, suppression needs to be considered for the threats that protect the target. One method of increasing the gain of an attack on a target is to conduct suppression of the corresponding threats; another method is to assign multiple packages to the same target. Define the maximum number of attack packages assigned to a target as the attack level and the maximum number of suppression packages assigned to a threat as the suppression level. Thus, different levels of attack can be made on each target. Similarly, different levels of suppression can be assigned to each threat.

In Model 1, only one level of attack for each target is allowed and no suppression of threat is considered. Now divide the targets into targets and threats. More precisely, we add threats to the model and define threats as both a target and a target defender. A threat is a target in its own right as well as a defender of more important targets. Each target has its own value, as does each threat. (This issue will be discussed in Section 3.) Based on the value of a target, one or more attack packages are chosen to attain sufficient destruction. (To match the requirement of attacking a target, a package is required to create a profit, i.e., destroying power, in some threshold.) By the same token, suppression packages are chosen for each threat. The degree of success in attacking the target is related to the level of suppression associated with the threats defending the target. The effectiveness of the attack packages assigned to a given target is reduced if the corresponding

defenders are not suppressed. The target damage effectiveness would be improved with higher levels of defender suppression. The effectiveness of the suppression packages assigned to a given threat is solely determined by the suppression itself and does not depend on attacking the defended targets.

To develop the suppression model, the following notation is needed:

D : the set of all threats (defenders),
S : the set of all possible threat suppression packages,
$[p]$: the set of $\{ 1, 2, \cdots, p \}$, where p is a positive integer,
S_d : a subset of S which consists of all suppression packages associated with threat d where the cardinality of S_d is $|S_d|$,
M : the highest desirable level of attack: Attack levels range from 1 to M where M is a positive integer and M is fixed for each target,
N : the highest desirable level of suppression: Suppression levels range from 1 to N where N is a positive integer and N is fixed for each threat,
$\bar{w}_{ta}$: the valid fraction of attacking power for attack package a on target t without considering suppression, where $a \in A_t$,
w^n_{tad} : the additional fraction of attacking power for attack package a on target t when threat d is suppressed at level n, where $a \in A_t$ and $\bar{w}_{ta} + \sum_{n\in[N]} \sum_{d\in D} w^n_{tad} = 1.0$,
g_{ta} : the profit associated with assigning attack package a to target t before evaluating suppression, where $a \in A_t$,
h_{ds} : the profit associated with assigning suppression package s to threat d, where $s \in S_d$ (Each threat and each target have their own values),
f_{cds} : number of class c asset contributing to suppression package s, where $s \in S_d$,
y_{ds} : equals 1 if suppression package s is assigned to threat d, and equals 0 otherwise, where $s \in S_d$,
δ^m_t : equals 1 if target t is attacked at level m or above, that is, there are at least m attack packages assigned to target t, and equals 0 otherwise,
θ^n_d : equals 1 if threat d is suppressed at level n or above, that is, there are at least n suppression packages assigned to threat d, and equals 0 otherwise.

The goal is to maximize the target and threat damage values subject to asset class assignment configurations and availability constraints. The model formulation is as follows:

Model 2

$$\max \sum_{t\in T}\sum_{a\in A_t}(\bar{w}_{ta}+\sum_{n\in[N]}\sum_{d\in D}\theta_d^n w_{tad}^n)g_{ta}x_{ta}+\sum_{d\in D}\sum_{s\in S_d}h_{ds}y_{ds} \tag{2.5}$$

$$s.t.\sum_{t\in T}\sum_{a\in A_t}e_{cta}x_{ta}+\sum_{d\in D}\sum_{s\in S_d}f_{cds}y_{ds}\le b_c \quad \forall c\in C, \tag{2.6}$$

$$\sum_{a\in A_t}x_{ta}-\sum_{m\in[M]}\delta_t^m=0 \quad \forall t\in T, \tag{2.7}$$

$$\sum_{s\in S_d}y_{ds}-\sum_{n\in[N]}\theta_d^n=0 \quad \forall d\in D, \tag{2.8}$$

$$\delta_t^{m+1}-\delta_t^m\le 0 \quad \forall t\in T, m\in[M-1], \tag{2.9}$$

$$\theta_d^{n+1}-\theta_d^n\le 0 \quad \forall d\in D, n\in[N-1], \tag{2.10}$$

$$x_{ta}\in\{0,1\} \quad \forall t\in T,\ a\in A_t, \tag{2.11}$$

$$y_{ds}\in\{0,1\} \quad \forall d\in D,\ s\in S_d, \tag{2.12}$$

$$\delta_t^m\in\{0,1\} \quad \forall t\in T,\ m\in[M], \tag{2.13}$$

$$\theta_d^n\in\{0,1\} \quad \forall d\in D,\ n\in[N]. \tag{2.14}$$

For each attack package a corresponding to target t ($a\in A_t$), the objective function (2.5) assigns a profit $\bar{w}_{ta}g_{ta}$, which represents the gain without considering the suppression of the corresponding threats, and assigns a profit $\sum_{n\in[N]}\sum_{d\in D}\theta_d^n w_{tad}^n g_{ta}$, which represents the marginal gains associated with the suppression of the corresponding threats. For each suppression package $s\in S_d$, the objective function assigns a profit h_{ds}, which represents the gain due to the suppression of the corresponding threat d itself. Notice that the coefficient for each suppression package is much simpler than the coefficient for each attack package. The gain associated with the assignment of an attack package to a target depends on the suppression of the target's defenders. But threats do not have defenders and, hence, the gain associated with the assignment of a suppression package on a threat is less complex. The objective function is nonlinear in this model formulation. This issue will be addressed later in this section.

Constraints (2.6) guarantee that whatever packages are chosen, the overall demand for each asset class will not exceed the supply. Constraints (2.7) guarantee that no target will be assigned more than M attack packages, while constraints (2.8) guarantee that no threat will be assigned more than N suppression packages. Constraints (2.9) enforce the precedence relation that higher levels of attack on a target will be considered only after all lower levels of attack on that target have been assigned. Similarly, constraints (2.10) enforce the precedence relation that higher levels of suppression of a threat will be considered only after all lower levels of suppression of that threat have been assigned.

Model 2 contains nonlinear terms (products of θ_d^n and x_{ta} for all $d\in D$, $n\in[N]$, $t\in T$, and $a\in A_t$). By introducing a continuous variable set z_{tad}^n, the fraction of additional gain of the attack package $a\in A_t$ associated with suppression of threat d at level n, the nonlinear objective (2.5) of

Model 2 can be transformed into a linear objective as

$$\max \sum_{t\in T}\sum_{a\in A_t} \bar{w}_{ta} g_{ta} x_{ta} + \sum_{d\in D}\sum_{s\in S_d} h_{ds} y_{ds} + \sum_{t\in T}\sum_{a\in A_t}\sum_{n\in[N]}\sum_{d\in D} w^n_{tad} g_{ta} z^n_{tad} \tag{2.15}$$

where z^n_{tad} is between 0 and $\min(\theta^n_d, x_{ta})$. That is, if there is no suppression of threat d at level n ($\theta^n_d = 0$), then z^n_{tad} will be zero and no additional value will be added into the objective function; if there is suppression of threat d at level n ($\theta^n_d = 1$), then z^n_{tad} will be bounded above by x_{ta}. As this is a maximization problem and the z^n_{tad}'s have positive coefficients, the values of z^n_{tad}'s will be at their upper bounds in the optimal solution. This characteristic enforces all z^n_{tad}'s to be 0 or 1 in any optimal solutions. Therefore, z^n_{tad} can be treated as a set of binary variables, and the whole model is a linear integer programming problem.

The linear transformation of Model 2 is listed below.

$$\begin{aligned}\max \sum_{t\in T}\sum_{a\in A_t} \bar{w}_{ta} g_{ta} x_{ta} + \sum_{d\in D}\sum_{s\in S_d} h_{ds} y_{ds} + \\ \sum_{t\in T}\sum_{a\in A_t}\sum_{n\in[N]}\sum_{d\in D} w^n_{tad} g_{ta} z^n_{tad}\end{aligned} \tag{2.16}$$

$$\text{s.t.} \sum_{t\in T}\sum_{a\in A_t} e_{cta} x_{ta} + \sum_{d\in D}\sum_{s\in S_d} f_{cds} y_{ds} \le b_c \quad \forall c \in C, \tag{2.17}$$

$$\sum_{a\in A_t} x_{ta} - \sum_{m\in[M]} \delta^m_t = 0 \quad \forall t \in T, \tag{2.18}$$

$$\sum_{s\in S_d} y_{ds} - \sum_{n\in[N]} \theta^n_d = 0 \quad \forall d \in D, \tag{2.19}$$

$$\delta^{m+1}_t - \delta^m_t \le 0 \quad \forall t \in T, m \in [M-1], \tag{2.20}$$

$$\theta^{n+1}_d - \theta^n_d \le 0 \quad \forall d \in D, n \in [N-1], \tag{2.21}$$

$$z^n_{tad} - \theta^n_d \le 0 \quad \forall t \in T,\ a \in A_t,\ d \in D, n \in [N], \tag{2.22}$$

$$z^n_{tad} - x_{ta} \le 0 \quad \forall t \in T,\ a \in A_t,\ d \in D, n \in [N], \tag{2.23}$$

$$z^n_{tad} \in \{0,1\} \quad \forall t \in T,\ a \in A_t,\ d \in D, n \in [N], \tag{2.24}$$

$$x_{ta} \in \{0,1\} \quad \forall t \in T,\ a \in A_t, \tag{2.25}$$

$$y_{ds} \in \{0,1\} \quad \forall d \in D,\ s \in S_d, \tag{2.26}$$

$$\delta^m_t \in \{0,1\} \quad \forall t \in T,\ m \in [M], \tag{2.27}$$

$$\theta^n_d \in \{0,1\} \quad \forall d \in D,\ n \in [N]. \tag{2.28}$$

The objective function (2.16) assigns a profit $\bar{w}_{ta} g_{ta} x_{ta}$, representing the gain without considering suppression of the corresponding threats, and assigns a profit $\sum_{n\in[N]} \sum_{d\in D} w^n_{tad} g_{ta} z^n_{tad}$, representing the marginal gains with suppression of the corresponding threats. For each suppression package $s \in S_d$, the objective function assigns a profit h_{ds}, which represents the direct gain of suppression of the corresponding threat d.

3. Development of Asset Allocation Problems. Important issues in designing realistic asset allocation problems were discussed in the *Real Time Strike Force Planning Workshop* in July 1995 at the Naval Research Laboratory in Washington, D.C. Based on concepts in Jakobovits (1995), we built the first model. Furthermore, we took the suppression issue into account and created Model 2. Because real problems were not obtainable for our computational research, we constructed an algorithm to generate problems that were as similar as possible to realistic problems. Two types of problems were created: semi-real problems, which represent possible military scenarios in the different regions of the United States, and randomly generated problems, which do not have associated topographical areas. The algorithms for building test problems for both models are similar to each other. For simplicity, we only outline the algorithm for the first model. A more comprehensive illustration for both models is available upon request.

For Model 1 all problems consist of two types of data: *asset classes* and *targets*; for Model 2, all problems consist of three types of data: *asset classes*, *targets* and *threats*. An *asset class* is defined as an asset type assigned to a particular location. Assets include carrier-based fighters and cruise missiles, targets include airports, metropolitan areas, bridges, railroads, military tactical headquarters, etc. The unit cost per asset (based on the type of asset) is assumed known.

Given the information about the asset types available and the unit cost per asset, a list of available options is then specified. Each option is a combination of different asset types, with the number of each asset type being specified (note that an option differs from a package in that packages are a collection of asset classes).

Each asset class has associated with it the (X, Y) map location, the cost per unit of asset type, and the capacity (number) of assets at this location. Similarly, each target has associated with it the (X, Y) map location, profit value associated with destroying the target, the number of different possible options available to destroy the target, and which particular options are available.

The semi-real and random problems differ in how this information is obtained and the differences are discussed further below. In the semi-real problems, the number of packages is fixed and all packages are included in the integer programming formulation. In the random problems, the size of the integer programming formulation is restricted by limiting the number of variables to be included. A percentage is specified which determines what proportion of the set of all qualified packages are to be included for each target. The inclusion of a specific package is on a random basis. To see the impact of asset capacity, we can vary the availability of resource capacity from very tight situations to relaxed conditions. On the other hand, we arranged the locations of the targets and the asset classes by taking the topography into account for the semi-real case. However, the topography issue was not incorporated in the randomly generated problems.

4. Computational Results. The computational platform used to solve the test problems consisted of the CPLEX Parallel Solver Version 6.6 (1999) on a Silicon Graphics Power Challenge with four 194 MHz R10000 processors and 1 GB of shared memory. The operating system was IRIX64 Release 6.5.

4.1. Results for Model 1. In Model 1, for each of the semi-real and random runs, a collection of three asset allocation problems were generated, with each problem based on one of three different engagement scenarios. Each problem can be distinguished by the number of asset classes and targets. Problem 1 consisted of 10 asset classes and 20 targets, problem 2 consisted of 20 asset classes and 40 targets, and problem 3 consisted of 30 asset classes and 60 targets. Problem 1 has 30 constraints including 10 knapsack constraints for the corresponding asset classes and 20 non-overlapping SOS constraints for the corresponding targets. Similarly, problems 2 and 3 have 60 and 90 constraints, respectively.

In the semi-real runs, only one instance was generated for each of the three problems. Note that a problem is distinguished by the number of asset classes and by the number of targets. The capacity of each asset class (the right-hand-side value of each knapsack constraints) and the number of packages (the number of variables included) are both fixed at the "instance" level. The number of packages and the capacity of each class of asset were fixed. Problems 1, 2, and 3, correspond to the Washington State map (Figure 1), the New York Metropolitan Area map (Figure 2), and the Northern California map (Figure 3), respectively.

Variations in the random runs were made as follows. To test the impact of increasing the number of packages (variables in the integer programming formulation) problem instances with 33%, 66%, and 100% of the potential packages were generated for each problem. In addition, the vector b, representing the number of assets available in each asset class, was varied to assess the impact of asset availability. In particular, all b_i were set equal to a single parameter β, where β was chosen to be 5, 10 or 15. While the b_i would not usually be equal in a realistic problem instance, our decision to fix b was intended to consider the cases of limited, moderately limited, and nearly unlimited asset availability through the choice of the single parameter. By allowing the package inclusion percentage and β to vary over three different values within each of three problems, we arrived at a total collection of 27 problem instances for the random runs.

In each instance for semi-real and random problems, the relative difference between the optimal LP relaxation value of the original problem and the best IP value found was calculated to indicate how closely the LP relaxation approximates the IP. Specifically, *lpipgap* was calculated as

$$lpipgap(\%) = [(\text{LP relaxation} - \text{best IP solution found})/\text{LP relaxation}] * 100\%. \tag{4.1}$$

We also calculated the relative difference between the best node remaining in the branch-and-bound tree and the best IP value found to indicate the solution quality. Note in a maximization problem, the best node is an upper bound for the MIP objective. In particular, the gap was calculated by

$$[(\text{best node} - \text{best IP solution found})/\text{best node}] * 100\%. \tag{4.2}$$

This gap is 0 if the optimal solution is found. Since the value of the LP relaxation is an upper bound of the value of the best node in the remaining branch-and-bound tree, the gap calculated by Equation 4.2 will be bounded above by the corresponding *lpipgap* calculated by Equation 4.1.

While CPLEX has a stand-alone MIP solver with default settings, it also provides a significant level of user control over parameter settings that direct the MIP solution process. With the default settings, decisions on whether or not to add cutting planes are made by CPLEX, and *mipgap* (the relative tolerance on the gap between the best integer objective and the best node remaining in the branch-and-bound tree) is 0.01%. For practical consideration, we relaxed *mipgap* to 1%. That is, the branch-and-bound process will terminate whenever the gap between the best node and the best IP solution found is less than or equal to 1%. In each integer programming run, the size of the branch-and-bound tree was limited to 1 GB of memory. To be practical, we also limited the run time for four processors to one hour, so all runs either reached a 1% gap solution, or the one hour run time limit was reached. Summary statistics for the semi-real problem instances of Model 1 are shown in Table 5.1, and Model 1 summary statistics for the random problem instances are shown in Tables 5.2 to 5.4.

For the nine instances of random problem 1, the performance is presented in Table 5.2. The column labeled *Simplex Iter.* represents the total number of simplex iterations needed to solve the problem, while the column labeled *No. of Nodes* represents the total number of nodes explored in the branch-and-bound tree. The column labeled *IP Obj.* gives the best integer solution found within the one hour time limit. The columns labeled *lpipgap (%)* shows the value calculated by Equation 4.1, indicating how close the original LP relaxation solution is to the best IP solution found. The columns labeled *Gap (%)* displays the result by Equation 4.2, indicating the quality of the corresponding IP solution. The run times (elapsed times) using four processors are shown in the column labeled *Run Time*.

In each instance of random problem 1 in Table 5.2, the original LP relaxation solution was very close (less than 2%) to the best solution found. Similar results were obtained for the nine instances of random problem 2 and the nine instances of random problem 3 in Tables 5.3 and 5.4, respectively.

The best solution found is within 1% of the best node for each instance in random problem 1. For random problems 2 and 3, the instances were much harder than those in random problem 1, especially when $\beta = 5$, and there were four instances which required more than one hour to converge to 1% of gap. However, the gaps of these four instances were all within 2%. Moreover, we found that solutions with good quality (10% of gap) were already available in the first minute of the solution process for these more difficult instances. The column labeled *MinuteGap(%)* in Tables 5.3 and 5.4 indicates the solution quality evaluated in the first minute during the run period. Good solutions, therefore, were not too hard to obtain in the first minute of the run even if it might take more than one hour or one day to get the optimal solution. Another observation is that the problems were relatively easy to solve when β was relatively large (when $\beta = 15$ as shown in the corresponding tables). This result can be explained in part by recognizing that as β becomes large, the constraints $Ax \leq b$ cease to be restrictive. The only constraints remaining are the mutually disjoint SOS constraints. In this situation, in each SOS constraint the variable with the largest coefficient in the objective function will be set equal to 1, and all others to 0.

Table 5.5 provides more detailed results for the collection of largest instances (100% of all eligible packages) of random problem 3. We extended the previous experiment by varying β over the values

$\{1, 2, \cdots, 15\}$, while fixing the package percentages at 100%. As can be seen from the test results, the gap between the optimal value of the original LP relaxation and the integer program is uniformly small except when $\beta = 1$. However, this problem is trivial to solve when β (asset class capacity) is so small since the majority of the variables (packages) can be eliminated from further consideration. The reduced problem becomes much smaller and easier to solve. Although the best solution found was not within 1% of the best node for the instances of $\beta = 3$, 5, and 6, the corresponding gaps were all less than 4%. Moreover, the best solution in the first minute for each β value was less than 10%. The state-of-the-art CPLEX software enables us to get *real time solutions* very close to the optimal solution for this problem class.

4.2. Results for Model 2. In Model 2, we generated a random problem with 10 asset classes, 10 targets and 10 threats. For variations for each problem, we also considered the number of packages and suppression packages with 33%, 66%, and 100% of the eligible packages and set β to 5, 10, and 15. Altogether we generated nine problem instances for the random runs.

Summary statistics for the random problem instances of Model 2 are shown in Table 5.6. The sizes of Model 2 problem instances are much larger than the problems from Model 1 due to the number of precedence constraints in Model 2. For example, there were 21,592 constraints for the 100% case of the random problem in Model 2, while there were only 90 constraints for the largest problem instance in Model 1. However, we were still able to reach the 1% gap within one hour for all nine problem instances. In fact, most of the instances reach the 10% gap in the first 10 minutes of run time as shown in the last column of Table 5.6. The only exception was the first instance with a gap of 12% in 10 minutes.

Table 5.7 provides more detailed results for the collection of largest instances (100% of all eligible packages) of the random problem. As can be seen from the test results, the gap between the optimal value of the original LP relaxation and the integer program was quite small for every instance except when $\beta \leq 3$.

Even though the best solution found was not within 1% of the best node for the instances of $\beta = 3$ and 4, both of the instances converged to a gap of less than 2%. Overall, the best solution in the first 10 minutes for each β value was less than 10% except when $\beta = 5$, 9 and 12. Although the size of this problem class is very large, we were usually able to obtain good solutions in a short period of time.

5. Conclusions. In this paper we have addressed the computational tractability of solving large asset allocation problems (without suppression and with suppression of defenders). Integer programming formulations were developed, and a broad variety of problem instances were generated to test the computational difficulty of the problems. The results demonstrated that in most instances, except when β value is very small, the original LP relaxation of the asset allocation problem (w/o suppression) is a very good approximation of the optimal solution. In those instances where optimality is elusive, feasible solutions very close to optimality can be generated in a short time period. In fact, good solutions are generated so quickly that it is a real possibility to consider using large integer programming models *during* the engagement in addition to the pre-engagement

planning period.

A number of important issues remain to be addressed. While every effort was made to generate realistic problem instances, working with real problem instances would have been preferable. For a variety of reasons, such data were not available at the time of our study. Even so, we believe our computational results are indicative of those that would be achieved on real problem instances for two reasons. First, the nature of the data encountered in real problem instances was established as a result of a sequence of workshops on strike force planning organized by the Navy and the generated problem instances were designed to have similar characteristics. Second, our computational results demonstrated that a wide collection of problem instances were uniformly easy to solve, suggesting that reasonable variations in the data do not make the problem significantly more difficult.

It would be of interest to incorporate routing in addition to asset allocation decisions. This extension would lead to more complex integer programming formulations, and these formulations may well be correspondingly more difficult to solve. However, the potential benefits from solving the expanded models would well be worth the effort.

Last but not least, we observe that CPLEX MIP Parallel Solver was able to solve the asset allocation problem w/o suppression sufficiently close to the optimal solution in a very short time frame. However, for some instances, it took a very long time to converge to our 1% goal. It would be interesting if heuristics could be designed to achieve this goal in a reasonable time frame.

Notation used in Figures 5.1,5.2 and 5.3

A	Airport	M	Metropolitan Area
AB	Air Force Base	MB	Military Base
B	Bridge	NB	Naval Base
C	Chemical Plant	R	Radar
H	Tactical Headquarters	S	Satellite Station
J	Aircraft Carrier	X	Railroad
K	Cruiser		

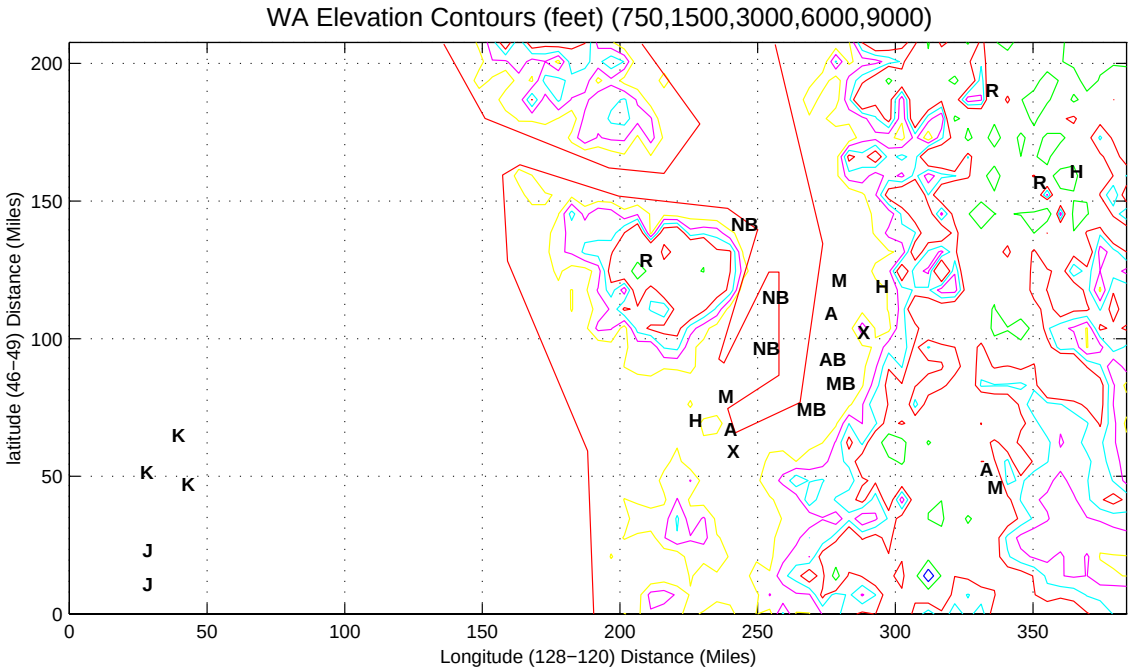

FIG. 5.1. *WA Map*

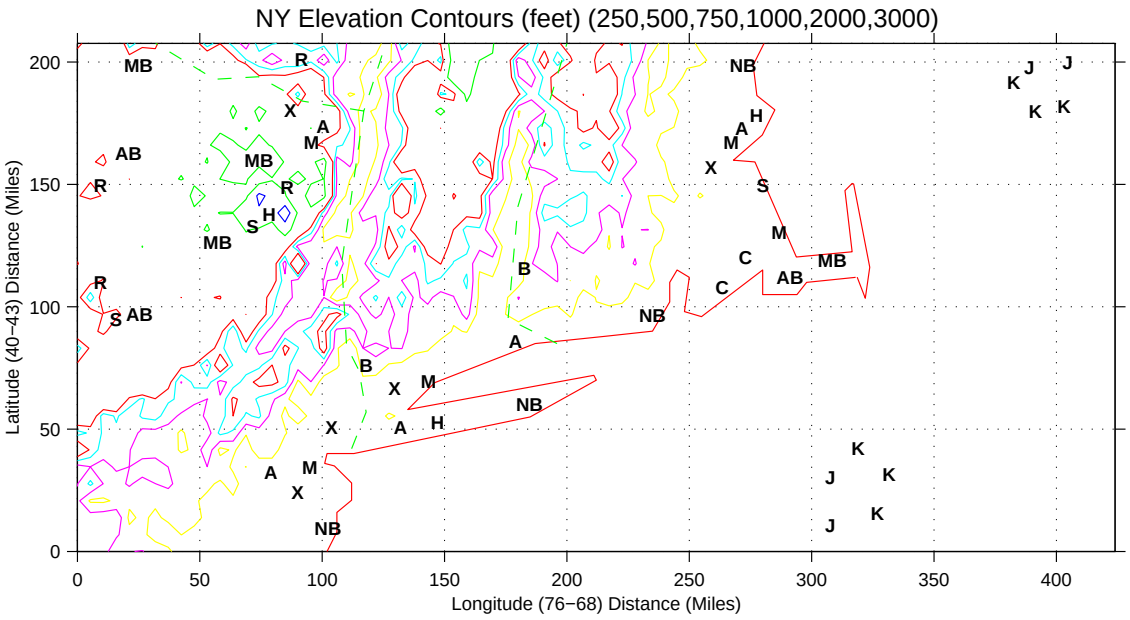

FIG. 5.2. *NY Map*

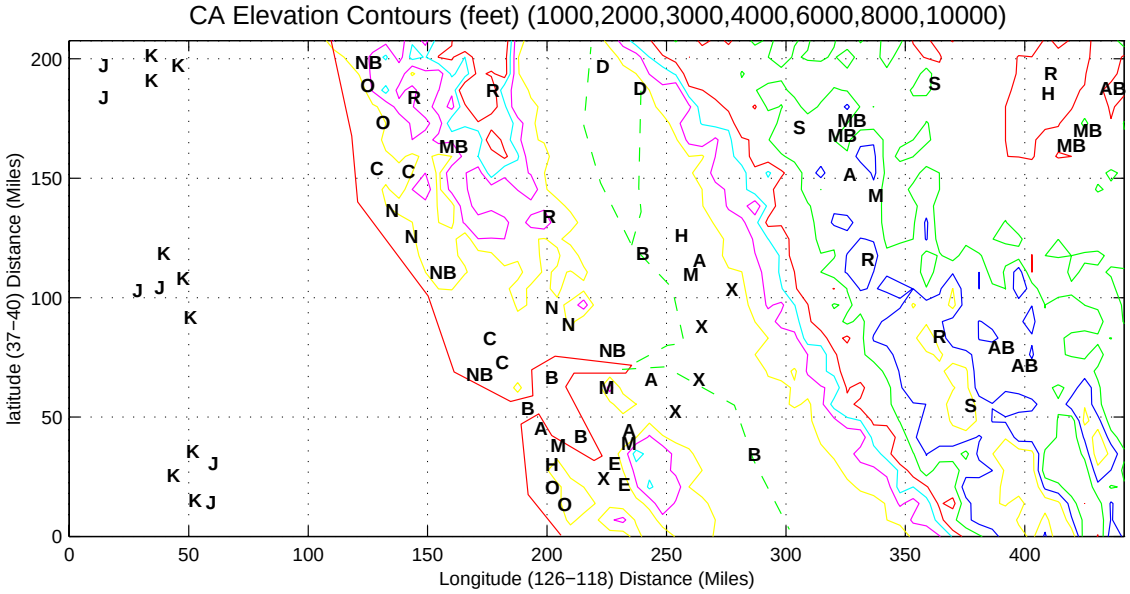

FIG. 5.3. *CA Map*

TABLE 5.1

Performance of Three Semi-Real Problems in Model 1 via CPLEX 6.6 Parallel Solver

Problem ID	No. of Vars.	No. of Cons.	Simplex Iter.	No. of Nodes	IP Obj.	lpipgap (%)	mipgap (%)	Run Time (sec.)
WA	678	30	248	0	199780	0.00	0.00	0.4
NY	1468	60	364	1	484151	0.07	0.05	1.4
CA	4347	90	582	1	1015870	0.12	0.12	4.8

TABLE 5.2

Performance of Nine Instances of Random Problem 1 in Model 1 via CPLEX 6.6 Parallel Solver

RHS (β)	No. of Vars.	Simplex Iter.	No. of Nodes	IP Obj.	lpipgap (%)	mipgap (%)	Run Time (sec.)
5		12467	1253	221470	1.96	0.90	22.9
10	3994	129	1	384476	0.99	0.92	6.4
15		210	1	447130	0.13	0.12	5.8
5		7640	750	221851	1.39	0.28	11.4
10	2645	978	89	383066	1.06	0.99	4.8
15		201	0	446650	0.00	0.00	2.6
5		1398	100	221421	0.86	0.17	2.7
10	1328	2653	195	380417	1.05	0.90	3.1
15		179	1	443334	0.02	0.01	1.3
number of constraints = 30 for each instance							

TABLE 5.3

Performance of Nine Instances of Random Problem 2 in Model 1 via CPLEX 6.6 Parallel Solver

RHS (β)	No. of Vars.	Simplex Iter.	No. of Nodes	IP Obj.	lpipgap (%)	mipgap (%)	Run Time (sec.)	1-min. mipgap (%)
5		2755404	147497	328356	1.90	1.73	$*$ 3600.2	2.95
10	8321	26302	883	623104	1.03	0.99	84.0	2.00
15		551	1	742640	0.06	0.05	16.2	0.05
5		4441622	242203	328406	1.81	1.61	$*$ 3600.1	1.97
10	5511	247112	8182	625571	0.61	0.54	291.1	0.86
15		492	1	741746	0.07	0.06	8.5	0.06
5		8426082	464712	327253	1.99	1.65	$*$ 3600.0	2.14
10	2765	284743	7805	623722	0.70	0.61	173.1	2.41
15		433	1	738595	0.21	0.21	3.8	0.21
number of constraints = 60 for each instance								
$*$ run time limit exceeded								

TABLE 5.4

Performance of Nine Instances of Random Problem 3 in Model 1 via CPLEX 6.6 Parallel Solver

RHS (β)	No. of Vars.	Simplex Iter.	No. of Nodes	IP Obj.	lpipgap (%)	mipgap (%)	Run Time (sec.)	1-min. mipgap (%)
5		1350509	48117	571340	1.54	1.12	$*$ 3600.3	1.92
10	15365	1016	15	1098985	0.27	0.23	52.2	0.23
15		1279	17	1374988	0.12	0.10	45.4	0.10
5		1431242	53484	571521	1.46	0.97	2508.7	4.52
10	10174	4276	159	1099138	0.20	0.15	34.7	0.15
15		929	1	1369247	0.36	0.35	20.8	0.35
5		1420299	50676	570207	1.52	0.93	1321.4	1.95
10	5102	27514	725	1087375	0.89	0.83	47.0	0.83
15		1270	36	1369088	0.09	0.08	10.0	0.08
number of constraints = 90 for each instance								
$*$ run time limit exceeded								

TABLE 5.5

Performance of Random Problem 3 in Model 1 with 100% Variables at Different RHS Values via CPLEX 6.6 Parallel Solver

RHS (β)	Simplex Iter.	No. of Nodes	IP Obj.	lpipgap (%)	mipgap (%)	Run Time (sec.)	1-min. mipgap (%)
1	148	1	95535	20.76	0.77	0.6	0.77
2	3274	391	235154	2.25	0.98	32.9	0.98
3	1503852	75192	339933	4.65	3.31	$*$ 3600.3	7.05
4	299509	15854	463886	1.25	0.90	1067.5	2.26
5	1350509	48117	571340	1.54	1.12	$*$ 3600.3	1.92
6	1077204	34381	676311	1.71	1.56	$*$ 3601.0	2.48
7	359954	12412	784595	1.19	0.97	1464.3	1.98
8	55790	1943	891112	0.88	0.78	295.8	2.20
9	330837	13155	991262	1.10	0.92	1239.3	2.58
10	1016	15	1098985	0.27	0.23	52.2	0.23
11	1911	31	1185117	0.90	0.85	61.8	0.85
12	71489	1358	1270873	0.70	0.64	318.7	2.21
13	20954	488	1329047	0.30	0.23	134.4	1.16
14	1401	12	1360195	0.01	0.00	49.5	0.00
15	1279	17	1374988	0.12	0.10	45.4	0.10
number of constraints = 90 for each instance							
number of variables = 15365 for each instance							
$*$ run time limit exceeded							

TABLE 5.6

Performance of Nine Instances of the Random Problem in Model 2 via CPLEX 6.6 Parallel Solver

RHS (β)	No. of Cons.	No. of Vars.	No. of Nodes	IP Obj.	lpipgap (%)	mipgap (%)	Run Time (sec.)	10-min. mipgap (%)
5			3826	172321	5.23	0.99	1336.8	12.06
10	21592	13851	1151	338974	1.98	0.54	847.1	3.53
15			840	484380	0.83	0.61	313.5	0.61
5			2803	171091	5.21	0.45	655.1	6.69
10	14304	9180	1164	333409	2.55	0.97	322.9	0.97
15			224	479243	1.00	0.81	68.0	0.81
5			2414	169479	5.59	0.96	259.3	0.96
10	7192	4620	1458	334668	2.30	0.97	143.7	0.97
15			2337	478625	0.91	0.68	98.1	0.68

TABLE 5.7

Performance of the Random Problem in Model 2 with 100% Variables at Different RHS Values via CPLEX 6.6 Parallel Solver

RHS (β)	Simplex Iter.	No. of Nodes	IP Obj.	lpipgap (%)	mipgap (%)	Run Time (sec.)	10-min. mipgap (%)
1	384	2	29466	25.49	0.00	0.4	0.00
2	19192	550	65226	14.64	0.42	255.8	0.93
3	438896	5283	99315	11.50	1.42	$*$ 3601.8	5.47
4	385144	9713	134706	8.75	1.48	$*$ 3602.1	8.83
5	144057	3826	172321	5.23	0.99	1336.8	12.06
6	364479	12783	207731	3.54	0.72	2678.3	4.60
7	124807	5785	241097	2.97	0.99	1941.9	6.86
8	101814	2622	275358	2.10	0.30	1262.5	0.76
9	121809	7859	306449	2.35	0.62	1889.3	12.91
10	39485	1151	338974	1.98	0.54	847.1	3.53
11	16592	37	372047	1.23	0.00	380.8	0.00
12	20284	561	399442	1.81	0.75	633.3	NA
13	59059	3799	426842	2.22	0.99	975.8	1.63
14	35392	4024	456758	1.63	0.86	1205.5	0.81
15	19519	840	484380	0.83	0.61	313.5	0.61
number of constraints = 21592 for each instance							
number of variables = 13851 for each instance							
$*$ run time limit exceeded							

ACKNOWLEDGEMENTS

The authors are indebted to Debra A. Elkins for providing useful writing comments on an earlier version of this paper. We would like to thank Ed Klotz for sharing his ample knowledge of CPLEX software. We also would like to thank Mark Reynolds and Donald L. Allen for providing useful comments on building up our test problems.

References.

Boroujerdi, A. (1994). *Joint Routing and Persistent Data Structures*. PhD thesis, University of New Mexico, Albuquerque, NM.

Boroujerdi, A., Dong, C., Ma, Q., and Moret, B. M. E. (1993). Joint routing in networks. *NET-FLO93: Network Optimization Theory and Practice*. Centro Studi Cappucini, San Miniato, Italy.

Cheong, C. K. (1985). Survey of investigations into the missile allocation problem. Master's thesis, Naval Postgraduate School, Monterey, CA.

Cho, C.-C. (2000). Optimal weapon allocation with target-dependant kill probabilities. Presented at the *Spring INFORMS National Meeting* in Salt Lake City, UT.

CPLEX 6.6 (1999). *Using the CPLEX Callable Library*. ILOG Inc. CPLEX Division, Incline Village, NV.

Eckler, A. R. and Burr, S. A. (1972). Mathematical models of target coverage and missile allocation. Technical Report DTIC: AD-A953517, Military Operations Research Society, Alexandria, VA.

G (1995). Real time strike force planning workshop. George Mason University, Fairfax, Virginia and the Naval Research Laboratory in Washington, DC.

Greenberg, H. J. (2001). *Mathematical Programming Glossary*. World Wide Web, http://www.cudenver.edu/ hgreenbe/glossary/glossary.html.

Hosein, P. A. (1989). *A Class of Dynamic Nonlinear Resource Allocation Problems*. PhD thesis, MIT, Cambridge, MA.

Hosein, P. A. and Athans, M. (1990). Some analytical results for the dynamic weapon-target allocation problem. Technical Report DTIC: AD-A219281, MIT, Cambridge, MA.

Jakobovits, R. H. (1995). Centralized and distributed resource optimization for strike warfare planning. Presented at the *Real Time Strike Force Planning Workshop* at George Mason University, Fairfax, VA and the Naval Research Laboratory in Washington, DC.

Lee, S. H. (1995). Route optimization model for strike aircraft. Master's thesis, Naval Postgraduate School, Monterey, CA.

Matlin, S. (1970). A review of the literature on the missile allocation problem. *Operations Research*, 18:334–373.

Zuniga, M. and Gorman, P. (1995). Threat site overflight modeling for strike route optimization. In *Proceedings of the First International Symposium on Command and Control Research and Technology*, Washington, DC. National Defense University.

Zuniga, M., Uhlmann, J. K., and Hofmann, J. B. (1994). The independence joint routing problem: Description and algorithm approach. Naval Research Laboratory, Washington, DC.