AFRL-IF-RS-TR-2004-237
Final Technical Report
August 2004

MODEL IDENTIFICATION AND OPTIMIZATION FOR OPERATIONAL SIMULATION

Systems View

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-237 has been reviewed and is approved for publication

APPROVED: /s/
GARY A. PLOTZ
Project Engineer

FOR THE DIRECTOR: /s/
JAMES W. CUSACK
Chief, Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Aug 04	Final Mar 03 – Feb 04

4. TITLE AND SUBTITLE
MODEL IDENTIFICATION AND OPTIMIZATION FOR OPERATIONAL SIMULATION

5. FUNDING NUMBERS
C - F30602-03-C-0167
PE - 62702F
PR - 459S
TA - MA
WU - 02

6. AUTHOR(S)
Douglas A. Popken and Louis A. Cox

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Systems View
9139 Roadrunner St.
Highlands Ranch, CO 80129

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFSB
525 Brooks Road
Rome, NY 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2004-237

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Gary Plotz, IFSB, 315-330-4383, plotzg@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

The purpose of this research effort was to develop and test frameworks and algorithms for use in air warfare planning systems. Developing planning systems for this problem domain is particularly challenging due to their great complexity and uncertainty. The effort focused on predictive simulation models for generating potential outcomes of proposed operational plans. The planning process was organized as a hierarchy of decisions, with those at the top being broadest and longest term. The algorithms at the highest level of planning use a hill-climbing approach, wherein proposed "Blue" plans are evaluated, and the average marginal benefits of alternative force reallocations are computed. Evaluation and measurement of each proposed Blue plan is accomplished via a "Stochastic Evaluator" that draws multiple samples of potential outcomes and "Red" force levels for a given Blue force structure and a combined target composition. The evaluation metric is the net discounted value from enemy targets hit. Within the evaluator, linear programming and simulation generate optimized Red responses, assumed outcomes, and relative marginal force values. This project successfully demonstrated automated plan optimization, practical embedding of optimization algorithms into an operational planning cycle operating over a multi-period conflict, and use of hierarchical decision-making to decomposed planning and on-line optimization problems into computationally practical tasks.

14. SUBJECT TERMS
Simulation, Optimization, Planning, Operational Simulation, Linear Programming, Hierarchical Planning, Air Combat, Uncertainty

15. NUMBER OF PAGES
56

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

Table of Contents i
List of Figures and Tables ii
Summary 1
  Problem Domain 3
  Previous Research 4
Methods, Assumptions, and Procedures 7
  General Algorithm 7
    Level 1: Allocating Forces to Roles 7
    Level 2: Assigning Forces to Targets 10
  Level 1 Mathematical Programming Formulation 10
    Discussion 12
  Attrition Rate Matrices 15
    Fire-Attribution Method 16
    Time-Over-Target Method 19
    Level 1 Approximation 19
  Update of AAF Target Values 21
  Creating a Level 2 Plan 22
    Discussion 24
  Stochastic Simulator 25
    Attrition Logic 26
    Target Engagement Model 28
Results and Discussion 29
  Database 29
  Verification Testing 29
  Performance Testing 29
    Level 1 30
    Level 2 31
  Face Validity 32
    Level 1 32
    Level 2 35
  Statistical Comparisons between Planned and "Actual" Conflict 35
Appendix A: AFSimPlan Software 41
  Overview 41
  User Guide 43
    Level 1 Analysis 43
    Level 2 Analysis 46
Appendix B: Domain Model 49
List of Symbols, Abbreviations, and Acronyms 51

List of Figures and Tables

Figure 1. Levels of Operational Planning for Air Warfare 3
Figure 2. Optimization/Simulation Scheme to Generate T-Period Outcome 9
Figure 3. Allocation of Fire for Offensive Forces at Target Location A 17
Figure 4. Allocation of Fire for Defensive Forces at Target Location A 18
Figure 5. MATLAB Benchmark for the Test Machine 30
Figure 6. Net Blue Plan Value vs. Improving Iterations (Horizon Weight=0) 33
Figure 7. Net Blue Plan Value vs. Improving Iterations (Horizon Weight = .50) 34
Figure 8. Model Use during a Planning Cycle 35

Table 1. Level 1 Performance Testing Summary 31
Table 2. Level 2 Performance and Face Validity Testing Summary 32
Table 3. Total Allocations By Role Over the Planning Horizon 32
Table 4. Value Extraction during a Conflict 36
Table 5. Discounted Net Blue Values Extracted with Varying Intelligence Quality 37

Summary

This research effort developed and tested frameworks and algorithms for use in air warfare planning systems. It focused on the use of simulation models as predictive tools for generating potential outcomes of proposed plans, and incorporating realistically high degrees of complexity and uncertainty into optimization of plans.

In this effort, the planning process is organized as a hierarchy of decisions, as follows:

- The top-most level, "Force/Roles", allocates forces across combat roles (*Counter Air*, *Air Defense*, *Target Reduction*, *AAF Suppression*, and *Other*) for the remainder of the planning horizon. This is the broadest, longest-term decision level.
- The next level down, "Targets/Missions", assigns forces to specific targets and missions, both offensive and defensive, in a given planning period.
- The third level, "Routes/Engagements", assigns specific routes and engagements to individual forces on an assigned mission.
- The bottom level, "Flight Control", addresses real-time piloting, navigation, and fire control.

This effort focused on developing and integrating the top two levels. The algorithms for the Forces/Roles allocation function at the top level (Level 1 planning) use a stochastic gradient (hill-climbing) approach to evaluate and improve upon proposed "Blue" plans using estimated average marginal benefits of alternative force reallocations. The evaluation metric is the net discounted value from hitting enemy targets. Plans are iteratively adjusted in the direction of increasing estimated marginal benefits until no improving reallocations can be identified.

Each proposed Blue plan is evaluated with a "Stochastic Evaluator". The evaluator repeatedly samples from a probability distribution describing the (uncertain to Blue) Red force levels. For each sample, the Stochastic Evaluator uses a succession of linear program based mathematical models to determine an optimal Red response to the Blue plan, conditioned on the current assumed Red force level. The mathematical models provide both the Red force allocation, comparable to Blue's current plan, as well as marginal relative resource values based on the linear programming dual variables or "shadow prices". For each period in the planning horizon, a separate mathematical model will develop a Red response for the subsequent periods. However, only the Red force allocation and dual variables relating to the first period of each model are retained. In this rolling optimization approach, a multi-period look-ahead strategy is used to plan future responses and to determine the best current (first-period) Red action. The future portion of the plan is discarded as new information becomes available, and plans are re-optimized in each period. Once the first-period decision has been determined, a single-period probabilistic simulation model simulates an outcome for forces lost and targets damaged. The surviving forces/targets become the assumed starting state for the next period, and the process is repeated until all periods in the planning horizon have been evaluated. Finally, the results from all of the samples are aggregated to generate a probability distribution of potential outcomes.

The algorithms for the Targets/Mission assignment function at the second level (Level 2 planning) similarly use a Stochastic Evaluator to generate a distribution of potential outcomes conditioned on Red's force levels. However, both base locations and the force quantities are included in the more detailed Level 2 plan since mission targets are being identified, and combat range becomes a factor. To generate specific Blue and Red target assignments as a function of the Level 1 role allocations, a greedy heuristic was developed to assign target-hitting aircraft on each side to the highest-value targets. Supporting forces for the target-hitting aircraft are then sequentially assigned on each side to maximize the probability of mission success. The Level 2 assignment algorithms can be used in stand-alone mode, but are also integrated into the Level 1 analysis to generate the detailed force deployments used by the simulation model.

A battery of tests measured the performance and validity of the resulting algorithms on a test dataset describing a notional Korean peninsula scenario. One set of tests demonstrated the ability of the hill-climbing algorithm to generate significant improvements to initial Blue plans. In other tests, plans with varying weights placed on survivability were generated by varying the horizon weight parameter. Placing more weight on survival resulted in greater allocations to the *Air Defense* role and away from the *Target Reduction* role. A final series of tests placed the planning algorithms in a decision cycle, similar to how they would be used in practice. Each cycle represented a period of a simulated conflict. Varying the maximum number of improvements per cycle showed how the plan optimization process adds value during the operational planning cycle. Poor intelligence on enemy forces demonstrably skewed the plan away from desired goals. In all of the tests, run-time performance remains a concern. With several minutes required to evaluate each plan, fully optimizing a plan over a 5-period planning horizon requires many hours.

This project successfully demonstrated the use of an integrated hierarchical planning as a framework for air warfare planning. It showed that linearized decision spaces can usefully approximate problems that would otherwise be intractable due to size and uncertain dynamics. Linear programming and simulation can operate in tandem over an extended planning horizon to help generate high-level plans tuned to the objectives of the planners. Uncertainty was handled by simulation and sampling. We showed how to automatically generate more detailed plans that are consistent with top-level plans, and that that effective plan optimization algorithms can be embedded in an operational planning cycle operating over a multi-period conflict.

Introduction

Problem Domain

Figure 1 outlines a hierarchical organization of military planning for air warfare, to be used as a reference framework. The levels are briefly described below.

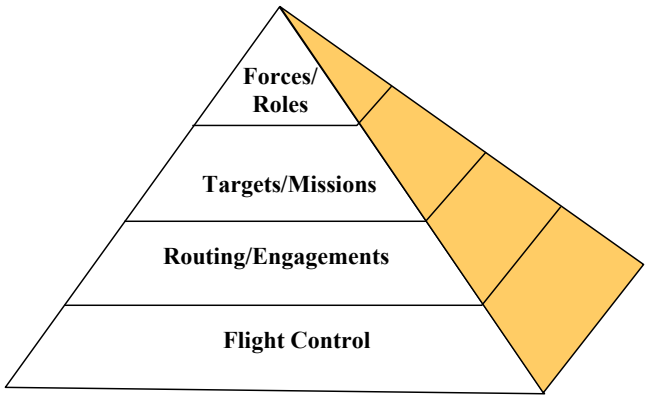


Figure 1. Levels of Operational Planning for Air Warfare

Level 1 - Assign Forces and Roles

The top level (Forces/Roles) sets force sizing and generic roles for different airframes over an extended planning horizon, and over a wide battle area. Roles include *Counter Air*, *Air Defense*, *Target Reduction*, and *AAF Suppression*. Many of the decisions at this level must be determined well in advance of any conflict. In this sense, this level precedes and hence lies above real-time operational planning. However, all military plans are subject to change once the conflict begins. Emerging intelligence changes the perceived enemy status, and event sequences often unfold in unexpected ways. It may become desirable to allocate more or fewer aircraft to their originally planned roles as time passes. The planning horizon at this level may be on the order of days or weeks to accommodate integration with overall war plans. Decisions from this level form an overall strategy for the conflict.

Level 2 - Assign Missions

The next level (Targets/Missions) determines specific targets for specific aircraft in a given attack wave. (For simplicity, flexibility and realism, air missions are organized into "waves" that may occur at a rate such as one per day or three per day.) Targets are defined generically as spatially localized tasks or activities to which air resources can be assigned. They may include defense of friendly targets as well as destruction of enemy targets, and may denote regions, weapon systems, or facilities. Thus, the different "roles" discussed above all are consistent with the concept of target assignment. A target list with defined target values or priorities is the basis of planning at this level. The planning horizon at this level is typically on the order of hours to

days to account for emerging targets and changing, frequently updated intelligence such as battle damage assessment.

Level 3 - Schedule Routes and Engagements

The third level (Routing/Engagements) determines specific routes for aircraft to follow from their base to their assigned targets and back. It can also determine what enemy forces should be engaged *en route* to the final objectives (targets). At this point, decision-making is influenced by the near real-time state of operations to include positions of specific enemy units. The planning horizon at this level is on the order of minutes to hours to account for force locations.

Level 4 Flight Control

The lowest level (Flight Control) refers to real-time piloting, navigation, and fire control. Decisions are on the level of detailed activities such as "jink-left" or "fire-missile". The planning horizon at this level is on the order of seconds to minutes. The flight control layer is not addressed in this project.

Appendix B specifies the detailed problem domain definition. The domain includes Forces (Aircraft and AAF), Missions, Orders of Battle, Targets, Locations, Plans, Planners, and Intelligence Estimates. The data used to populate the domain model came primarily from two sources. The first was a spreadsheet dataset used by AFRL/IFSB for model testing known as "the Korean Scenario". This provides data for a subset of US forces in and around South Korea. The second was a well known public web site (http://www.globalsecurity.org), providing data on North Korean forces and bases.

Previous Research

Early models for planning and optimizing in air combat focused on simplified stochastic games and differential games (Isaacs, 1965). Karlin (1959) discusses the application of dynamic programming, linear programming, and network flow analysis to the problem. Dresher (1981) provides a number of interesting and still relevant formulations on topics such as the "Tactical Air-War Game", which we used for a simplified Level 1 (Forces/Roles) allocation problem, and "Defense of Targets of Many Values", which has useful implications for our Level 2 (Targets/Missions) analysis.

Significant research, largely within the DARPA JFACC (Joint Force Air Component Commander) program (Heise and Morse, 2000), has investigated planning of military operations near the third or even fourth level (Routing/Engagements or Flight Control) discussed above. Many of these efforts emphasized a control-system based mathematical framework. Mukai et al. (2000) provides a differential game formulation for opposing air and ground units. Unit movement (on a 2-D grid) and attrition are governed by differential equations. The controls are the speed, direction, and engagement "intensity" of each unit in each time period. The value of the game is determined by a quadratic payoff function that tends to: 1) minimize distance

between units and their designated target locations, and 2) maximize enemy losses for any given level of friendly losses. Cruz, et al. (2002) formulate a similar problem where the unit state describes location, the number of "platforms", and the number of weapons per platform. The controls are movement, salvo size, and choice of target. The objectives are again to maximize enemy losses while minimizing friendly losses. McEneaney and Ito (2000) attempt to optimize both aircraft routes and engagement decisions made against hostile missile units and fixed strategic targets. The unit state describes location (for aircraft and mobile missiles) and "health". The objective is to destroy strategic targets while each side tries to minimize its own losses and maximize its opponent's. The controls are determined via a two-level hierarchical optimization procedure. First, the aircraft routing through the hostile region is determined; next, the time ordering of aircraft engagements with missile sites is determined.

Each of the JFACC efforts above discretizes the decision space, creating a "curse of dimensionality" for combinatorial optimization algorithms. They also assume perfect knowledge of the true system state at all times. Finally, some may involve decisions best left – at least for now – to humans on the scene. (For a fascinating, and still relevant, though controversial, discussion on human *vs.* computer decision-making, see Dreyfus - 1994) A more recent paper by McEneaney et al. (2003) refocuses their earlier model on *unmanned* combat air vehicles and points out that the formulation provided is for lower levels of a decision hierarchy. He suggests that the highest level decides strategic planning and resource allocation; a middle level determines aircraft routes, engagement assignments, and support activities; while the lowest level determines targeting and vehicle guidance. The paper also introduces the notion of having only partial information on the system state. In a final nod to reality, the paper recognizes the severe problem size limitations imposed by combinatorial explosion.

Cave and Busch (2003) also take a control-system approach to formulating the operational planning problem within an Aerospace Operations Center. They provide examples of how the system could work at the third level (Routing/Engagements) for aircraft being routed to targets across a hostile battlefield. The system explicitly accounts for uncertainty, but modeling of Red counter-action dynamics is limited. The existence of higher decision making levels is mentioned, but an integration framework is not discussed.

The algorithms developed and tested in this project build on the preceding ideas and extend them with several related advances in decision sciences. The main additional concepts from the literature that were incorporated into our integrated hierarchical planning approach are as follows.

- *Simulation-optimization* (Carson and Maria, 1997; Dippon, 2003). Stochastic gradient (hill-climbing) techniques useful for simultaneously *estimating* local response surface shapes (i.e., how does the expected value to Blue change with alternative Blue decisions?) from samples generated by probabilistic simulation, and *adjusting* Blue's decisions to maximize expected value, have a long tradition in what is now called simulation-optimization. These ideas reach back to the Evolutionary Operations (EVOP), Kiefer-Wolfowitz, and Robbins-Munro iterative adjustment procedures for climbing unknown response surfaces. Our approach exploits simulation-optimization techniques to automatically improve Blue's plan, while

adding the novel feature of an embedding Red's best response strategy (by "optimizing out" Red's choices via a multi-period linear programming approximation).

- *Game theory and linear programming.* From classical game theory (e.g., Dresher, 1981) we adopted the idea of letting Red adopt a best response to Blue's plan, thus encouraging Blue to identify a traditional minimax strategy (when one exists in pure strategies). In effect, this assumes that Red will have excellent intelligence and respond as if well informed about Blue's plans. If this perspective turns out to be overly pessimistic, Blue's plan will be re-optimized immediately (in one period) to take advantage of Red's sub-optimal play. We exploited the idea of shadow prices and hierarchical (Dantzig-Wolfe) decomposition from large-scale and decentralized linear programming to coordinate among levels of the decision hierarchy and to help optimize out Red's planning decisions. In a departure from traditional game theory, solutions were required to be pure strategies, i.e., randomized plans were ruled out as unrealistic to implement, despite their potential theoretical advantages. (This entails no loss of generality if the pure strategy sets are convex and each player's payoff function is continuous in both Red's and Blue's pure strategies and is quasi-concave in its own pure strategies. But such nice mathematical results are not available for the discrete allocation optimization choice sets of practical interest in real conflicts; hence we simply enforced a no-randomization constraint.) Finally, we applied ideas of *mechanism design* from modern game theory, to formulate the conflict as a bi-level hierarchical optimization problem, in which Blue chooses a plan assuming that Red will choose a best response to it. This avoids the need to find (probably unrealistic and computationally daunting) Bayesian Nash equilibria, and reduces the problem to one that can be solved by adaptive optimization methods (with suitable technical precautions, such as the inclusion of tabu-search type restrictions (Glover and Laguna, 1998) to prevent hill-climbing optimization routines from cycling.)
- *Reinforcement learning and adaptive optimization.* In a real air conflict "game", neither player necessarily knows the structure or data of the games – e.g., the payoff functions and the transition functions in Markov games (Lagoudakis and Parr, 2002). Common-knowledge priors from which to compute theoretical Bayesian Nash equilibria cannot be justified. To model such realistically incomplete knowledge, we assumed that players can only sample from the distribution of outcomes for different pairs of Red and Blue decisions, via simulation. Thus, the players are treated as both *learning about* the payoff function and *adaptively optimizing* their decisions as the game evolves. Our models and algorithms therefore combine ideas from adaptive dynamic programming, especially multi-period rollout heuristics (Bertsekas et al., 1997) for solving Red's multi-period optimization problem, with "learn as you go" ideas from reinforcement learning. Rollout heuristics take a simple decision rule (e.g., based on an LP model, in our case) and apply it several steps ahead to forecast returns over a finite look-ahead horizon. Then, they re-optimize the initial (simple) decision using the forecast returns as an approximate value function. Reinforcement learning estimates value functions for state-specific decisions from sample data (Lagoudakis and Parr, 2002) and adjusts the decision rules to improve expected values. Both sets of ideas play essential roles in the algorithms and heuristics that follow. We apply them directly to the conflict simulation, rather to a simplified Markov game, to make the solutions as realistic as possible.

Methods, Assumptions, and Procedures

General Algorithm

Level 1: Allocating Forces to Roles

Level 1allocate Blue's forces to roles. We use the following roles for purposes of illustration, although the method is general: *Counter Air*, *Air Defense*, *Target Reduction*, *AAF Suppression*, and *Other*. Appendix B summarizes the input data, and the following sections provide greater detail where needed. Level 1 allocates a fraction of each type of Blue force to each role for each time period in the planning horizon. An allocation can also be interpreted as a plan or as a high level "Course of Action" (COA).

The following Blue plan improvement algorithm accomplishes this allocation:

Blue Plan Improvement Algorithm

INPUTS: An initial feasible plan for Blue; initial values for the parameters (ω, α, ε) = (horizon weight, discount factor, force reallocation proportion step size); D_1 = the starting problem domain data (including probability distribution for Red's forces, targets, etc)

OUTPUTS: An array PB_{ijt} giving the fraction of Blue's force of type i resources allocated to role j in period t, for each t in the planning horizon. This is a Level 1 *plan* for Blue.

1. **Initialize the horizon weight (ω = .50), the discount factor (α = .80), and the force reallocation proportion (ε = .10), to values between 0 and 1. Set the planning iteration index z =0. Initialize using an initial feasible Blue plan. This is an array of non-negative allocation fractions, $PB_{ijt}(0)$ for each force *i*, role *j*, and time period *t* (t=1,2,…T) subject to $PB_{ijt}(0) \geq 0$, $\sum_j PB_{ijt}(z) = 1, \forall$ i, j, t. (This initial plan may be generated randomly, via a heuristic, or imported from an external source.)**
2. **Given $PB_{ijt}(z)$, simulate the probability distribution of optimized Red responses and outcomes using ω and α along with the starting problem domain data (forces, targets, etc), D_1. This will provide a mean and a standard deviation, $\bar{\pi}_{ijt}(z)$ and $s_{ijt}(z)$, of relative marginal value for each Blue force, role, and time period (more details below).**
3. **Use $\bar{\pi}_{ijt}(z)$ and $s_{ijt}(z)$ to modify Blues's current plan to improve Blue's expected objective function. In general, we seek the reallocation of force where the net, statistically significant gain, is greatest.**
4. **Adjust Blue's allocations in the direction indicated, by adding ε to the gaining role allocation and subtracting ε from the losing role allocation (if the losing role**

allocation is currently less than ε we use that proportion instead) to obtain $PB_{ijt}(z+1)$. Set z =z + 1.

5. **If no further statistically significant improvements are found, STOP, otherwise return to Step 2**

In Step (1), default values of the input parameters are provided in parenthesis. It is necessary to introduce a list of remembered "tabu" elements (Glover and Laguna, 1998) to the plan improvement search in Step (4) to prevent a reallocation from being reversed for some number of improving steps, thus preventing cycling.

Step (3) of the plan improvement algorithm generates ***K*** sample values for each incumbent plan. Each individual "sample" outcome is a function of an instance of the possible Red force states. The Red force state is drawn from a probability distribution that is assumed to be a result of the latest intelligence estimates. At Level 1, the outcome is generated according the following general algorithm:

Blue Plan Stochastic Evaluator (Level 1)

For k = 1 to K

Set the estimated Red force levels by sampling from their probability distribution based on $D_1(k)$. Set $V_k(z)$ = estimated net Value to Blue of plan $PB_{ijt}(z)$ to 0.

For t = 1 to T

a) **Determine Red's approximate best response in period t by optimizing over the next T – t + 1 time periods using $D_t(k)$ and the horizon weight, ω. Select the first period of the solution as Red's response in period t, obtaining both Red's role allocation, $PR_{ijt}(z)$, and Blue's relative marginal resource values, π^k_{ijt}.**
b) **Using $PB_{ijt}(z)$, $PR_{ijt}(z)$, and $D_t(k)$, create Level 2 plans for period t for both Blue and Red.**
c) **Use the Level 2 Plans to determine target assignments.**
d) **Use the target assignments to allocate forces, and run the stochastic simulator for 1 period.**
e) **Set the Red and Blue force and target states to those found at the end of the one-period simulation, and update the problem data: $D_t(k) \rightarrow D_{t+1}(k)$**
f) **Set $V_k(z) = V_k(z) + \alpha^{t-1} V_{kt}(z)$ where $V_{kt}(z)$ is the observed net value to Blue extracted from targets during the period t simulation.**

END

END

Compute $\bar{\pi}_{ijt}(z) = \frac{\pi_{ijt}^{k}}{K}$ **= and** $s_{ijt}(z) = \frac{\sum_{k=1}^{K}\left(\pi_{ijt}^{k} - \bar{\pi}_{ijt}\right)^2}{(K-1)}$**, the average and standard deviation over K iterations for the** π_{ijt}^{k} **returned by Step (a).**

The outputs of the algorithm are the $\bar{\pi}_{ijt}(z)$, $s_{ijt}(z)$, and the $V_k(z)$. Note that the discounted Blue net values, $V_k(z)$ are not used in the optimization algorithm, but can be examined to view its performance or to compare alternative improvement algorithms (as in the "Testing" section).

The approach used in the Stochastic Evaluator is illustrated in Figure 2 below. The long-term optimization will capture long-term effects. The short-term response is then guided by the long-term response. The short-term simulations capture uncertainty.

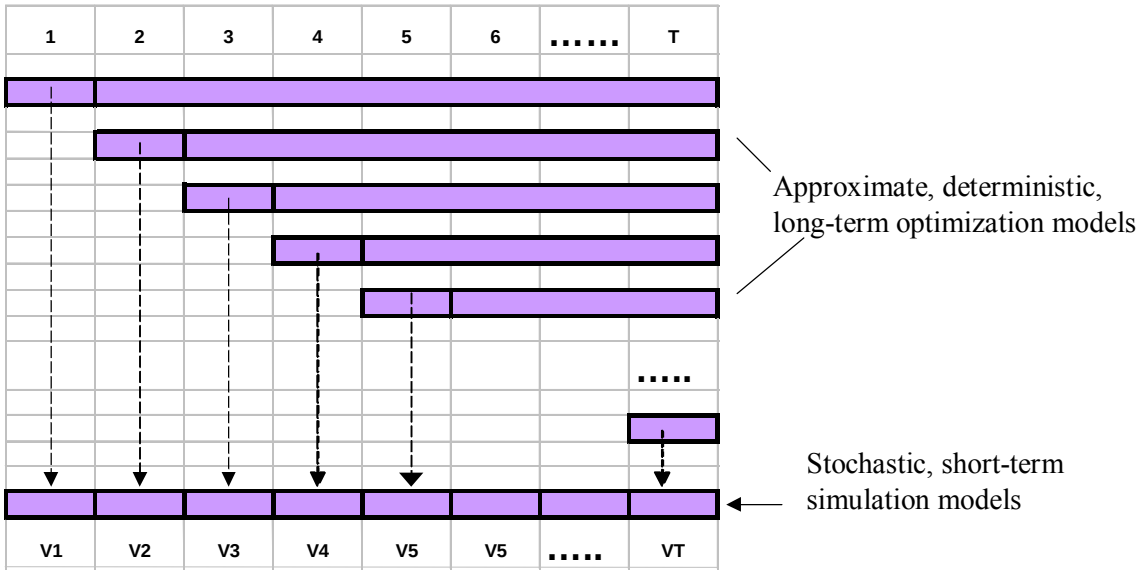


Figure 2. Optimization/Simulation Scheme to Generate T-Period Outcome

The horizon weight, ω, and the discount factor, α, work in tandem. The horizon weight places a value on survival of forces beyond the T period planning horizon. A short term perspective, ω = 0, implies that forces have no value beyond T. As a result, the relative marginal resource values for forces will be skewed in favor of the *Target Reduction* Roles. A long-term perspective, ω = 1, implies that survival is given high consideration. Forces are conserved at the expense of more immediate target value extraction. The discount factor, α, on the other hand, is not a control parameter, but an evaluation parameter. It acknowledges that, other things being held equal, we prefer shorter conflicts to longer ones. Target value extraction occurring early is more valued than the same target value extraction occurring later.

The optimization technique used in Step (a) is described in further detail below in the section, "Level 1 Mathematical Programming Formulation. Step (b) is described below in the section "Creating a Level 2 Plan". The stochastic simulation of Step (c) is described further under "Simulation Model".

Level 2: Assigning Forces to Targets

Level 2 is the lowest level modeled in this project – no downward integration was required. Many of the steps required for a Level 2 analysis are performed automatically at Level 1, since a detailed Level 2 plan is needed to simulate the outcome of the corresponding Level 1 allocation. To give the user some control over the target assignment process of Level 2, it must be possible to evaluate alternative plans. That is the role of the Level 2 Stochastic Evaluator described in this section.

Blue Plan Stochastic Evaluator (Level 2)

If automated target selection is desired, initialize Blue's target set to Null. Initialize the domain data D_1 to include the target set: D_1(BlueTargetSet)

For k = 1 to K

1) **Set the estimated Red force levels by drawing an instance from their probability distribution: →D_1(k, BlueTargetSet)**
2) **Using PB_{ijt}, PR_{ijt}, and D_1(k,BlueTargetSet), create Level 2 plans for both Blue and Red.**
3) **Using the Level 2 Plans to make target assignments, run the stochastic simulator for one period to obtain, V_k, the net value to Blue extracted from targets during the k^{th} simulation.**

END

The primary output is the V_k. The V_k can also be recomputed as an average over the targets selected as shown in the AFSimPlan User Guide

Level 1 Mathematical Programming Formulation

This section describes a method to optimize Red force allocation over T periods, given a T-period Blue force allocation (i.e., a Level 1 Plan). The Blue allocation is determined by the current Blue plan. The approach also returns Blue's relative marginal values of resources in the form of linear programming shadow prices. To enable a linear programming approach, constraints on force quantities are relaxed to allow fractional forces. A further relaxation on Red attrition is described and limitations of the technique are identified.

In each period, t (t =1, 2,…T), surviving forces on both sides are first allocated to some combination of roles: {*Counter Air, Air Defense*, *Target Reduction*, *AAF Suppression*, and *Other*}. Attrition then occurs in each force/role combination according to rules regarding which offensive and defensive roles may oppose each other. These rules are encapsulated in attrition rate matrices, described below. Forces in the *Other* role in a given time period do not participate in combat activities and do not undergo significant attrition[1]. The force types in the Level 1 model correspond to the different airframes and AAF types. AAF can only fulfill the *Air Defense* and *Other* roles. All force types can be assigned to the *Other* role.

variables:

ra_{ijt} = # of Red forces of type i allocated to role j at the beginning of t; i=1,2,…F_r; j = 1-5
ba_{ijt} = # of Blue forces of type i allocated to role j at the beginning of t; i=1,2,…F_b; j = 1-5
r_{ijt} = # of Red forces of type i, surviving period t attrition, that were allocated to role j at the beginning of period t
b_{ijt} = # of Blue forces of type i, surviving period t attrition, that were allocated to role j at the beginning of period t

Assume ***j*** = 3 represents the *Target Reduction* role. Only forces assigned to this role can directly "extract" value by destroying enemy targets.

r(t), **ra(t)**=row vectors of length F_r*5 of Red force quantities (all i,j) in period t
b(t), **ba(t)**=row vectors of length F_b*5 of Blue force quantities (all i,j) in period t

input parameters:

- PB_{ijt} = Blue's weights for force i, role j, in period t (these sum to 1.0 across a given (i,t) - this is the current Blue conceptual plan)
- LAMBDAB = rate at which Blue forces (i,j) destroy Red forces (i',j') (2-d matrix)
- LAMBDAR = rate at which Red forces (i',j') destroy Blue forces (i,j) (2-d matrix)

The LAMBDAx matrices reflect the invulnerability of the *Other* role.

- vb_i = average value reduction (from red's perspective) that Blue force i in the *Target Reduction* role achieve
- sb_i = salvage value factor for Blue force i
- vr_i = average value reduction (from red's perspective) that Red force i in the *Target Reduction* role achieve
- sr_i = salvage value factor for Red force i
- vrTotal = the total remaining value of all Red targets (Red perspective)
- **b(0)** = any initial feasible assignment of current Blue forces to roles
- **r(0)** = any initial feasible assignment of current Red forces to roles

Optimizing from Red's perspective, the problem is initially formulated as P(T).

[1] Their attrition rate may actually be a relatively small nonzero value that does not have a major impact on high-level strategy.

P(T):

Maximize $\sum_{i=1}^{F_r}\left[\sum_{t=1}^{T}(vr_i)r_{i3t}+\sum_{j=1}^{5}(sr_i vr_i)r_{ijT}\right]-\sum_{i=1}^{F_b}\left[\sum_{t=1}^{T}(vb_i)b_{i3t}+\sum_{j=1}^{5}(sb_i vb_i)b_{ijT}\right]$

Subject to:

max value obtainable:

$$\sum_{t=1}^{T}\sum_{i=1}^{F_r} r_{i3t} vr_i \le vrTotal$$

expected attrition: t = 1, 2, 3.... T

b(t) = max(**0**, **ba(t) - ra(t)***LAMBDAR)
r(t) = max(**0**, **ra(t) - ba(t)***LAMBDAB)

Red reallocation: t = 1, 2,....T

$$\sum_{j=1}^{5} I_{ij} ra_{ijt} = \sum_{j=1}^{5} r_{ij(t-1)} \quad i=1,2,\ldots F_r \quad (I_{ij}=1 \text{ if role j is feasible for force i; 0 otherwise})$$

Blue reallocation to plan: t = 1,2,....T

$ba_{ij't} = PB_{ij't} * \sum_j b_{ij(t-1)}$ i=1,2,…F_b; j' = 1,2,3,4,5

nonnegativity:

$ra_{ijt}, r_{ijt}, ba_{ijt}, b_{ijt} \ge 0$

Discussion

Note that the Red allocation proportions, $PR_{ijt}(z)$, discussed in the previous section are determined by $PR_{ijt}(z) = \frac{ra_{ijt}}{\sum_j ra_{ijt.}}$. The Blue relative marginal values for resources, π_{ijt}^{k}, are the values of the dual variable, i.e., "shadow prices" on the "Blue reallocation to plan" constraints.

This is a conservative formulation in that aircraft must survive the entire mission to extract target value. In the simulation, aircraft must only survive the first ½ of the mission to extract target value.

Forces surviving until period T are given a salvage value equal to $sb_i * vb_i$ (for blue) or $sr_i * vr_i$ (for red). This reflects the fact that forces have value beyond the scope (planning horizon) of the model.

To solve this formulation via linear programming, eliminate the nonlinear "max" in the attrition dynamics for Blue by replacing the attrition equations with:

$$\mathbf{b(t) \geq ba(t) - ra(t)}*\text{LAMBDAR} \qquad t = 1, 2, \ldots T$$

There is no incentive for b(t) to be large since the Blue variables only enter the objective function with negative sign. In most cases, there is a *disincentive* for them to be large, so that the left hand side will be set to zero when the right hand side is negative. In the other cases, the value of b(t) will not matter since there is no effect on the objective function. An example would be the numbers of Blue forces allocated to non target reducing roles in period T (b_{ijT}, $j \neq 3$)

This approach would not work for the Red variables, since they enter the objective function with positive sign. The problem would be unbounded. In principle, one could enforce the Red attrition constraints by introducing binary variables, $y_{ijt} = 0,1$ such that:

$$\mathbf{r(t) = y(t)}*[\mathbf{ra(t) - ba(t)}*\text{LAMBDAB}] \quad t = 1, 2, \ldots T$$

But this formulation would not be computationally practical to solve directly as there are many binary variables, and the constraints are nonlinear. The reallocation at each period further complicates the solution; an airframe/role combination that goes to zero in one period may become nonzero in the next through a reallocation. The y(t) must also be constrained to make physical sense, i.e., once an entire force type has been eliminated, the corresponding y_{ijt} must remain at 0 for subsequent t. We instead first solve the relaxed problem using:

$$\mathbf{r(t) = ra(t) - ba(t)}*\text{LAMBDAB}; \; -\infty \leq \mathbf{r(t)} \leq \infty \qquad t = 1, 2, \ldots T$$

to represent Red's attrition, while eliminating the nonnegativity constraints on the r(t). Refer to this relaxed formulation as **RP(T).** Solutions to **RP(T)** have several notable characteristics:

- Red can have attrition in an (i,j,t) combination even if there are no Red forces allocated there. This will result in a negative value for the corresponding element of r(t).
- This "fictitious attrition" will be deducted from the quantity that can be allocated in period t+1.
- It is possible that no feasible solution can be found. If all units of a given force type are destroyed by real or fictitious attrition by period t, there is no way to allocate positive quantities, ra(t+1). However, we must retain the lower bound of 0 on the allocated quantities, otherwise the LP will allocate negative quantities to the invulnerable *Other* role to pump up the allocation to other roles.
- The solution to **RP(T)** will provide a lower bound on the objective function (value to red) of **P(T)** since Red may have higher attrition in the solution than is warranted by the allocations. (Note that negative Red values will decrease attrition on blue, perhaps to the point of having negative attrition.)
- The infeasibilities in this approach increase as T increases since the negative attrition is cumulative.

We have developed a practical heuristic to generate a good feasible (but not necessarily optimal) solution to **P(T)** that accounts for the above considerations. Because of this, and the fact that we are modeling stochastic attrition and target reduction with average deterministic values, *we will only extract the period 1 solution for Red's allocation.* Similarly, *we will only extract the shadow prices on Blues period 1 allocation*. But for these values to capture long-term effects, *we still must extend ("roll out") the model out to T periods*. We proceed with the following algorithm to ensure a T period feasible solution:

Generating Good Feasible Solutions to P(T)

1. **For t' = 1,2,…T**
    a. **Solve RP(t')**
    b. **Refer to the Red solution variables as r_{ijt} and ra_{ijt} (post-attrition variables and allocation variables). For each post attrition variable $r_{ijt} < 0$, t=1,2,…t', add a constraint to the LP such that $r_{ijt} = 0$.**
2. **solve RP(T)**

This algorithm requires solving (T+1)(T+2)/2 – 1 linear programs for a single sample of the Level 1 Stochastic Evaluator.

It may also be possible to devise a more efficient branch and bound scheme to find a globally optimal solution, but this is beyond the scope of the current effort.

The above algorithms yield the proportion of aircraft Red allocated to each role in each period. We can compute CR_{ijt} from these to get the Red equivalent to Blue's plan. As noted above, the optimality of the Red plan is more suspect for higher values of t, but only the first period allocation is used, even though the LP is run for T periods in an attempt to estimate longer term effects, consistent with the "rollout" heuristic principle (Bertsekas et al., 1997). This approach was also explored in Yost and Washburn, who applied it to attack aircraft hitting targets in stages.

The LP model also provides prices for the Blue allocation, indicating where Blue could benefit by adjustments to the plan. The prices are the dual variables of the "Blue reallocation to plan" constraints. Blue's value is obtained by recomputing the objective function using v's for Blue.

The formulation is conservative in that it assumes that target-reducers (i.e., hitting or striking planes)must survive the period to obtain target value in that period.

In the domain model, ***TargetEffectiveness*** is the average capability degradation per strike, by target, when this aircraft is in the *Target Reduction* role. We represent this using E_{ik}, the effectiveness of airframe i when used against target k. The domain attribute, ***RedValue,*** is the value that Blue believes that Red currently places on a target. Represent this using v^R_k for targets on Reds target list (Blue assets) and v^B_k for targets on Blues target list (Red assets)

$$vr_i = \frac{\sum_{k=1}^{K^R} E_{ik} v_k^R}{K^R} \qquad vb_i = \frac{\sum_{k=1}^{K^B} E_{ik} v_k^B}{K^B}$$

In practice (and in the simulation model), each side will likely strike the targets in order of decreasing marginal expected value. If we were modeling Red only, we could include this feature using concave piecewise costs in the objective function, automatically forcing Red to hit the highest value targets first. However, this will not work for Blue since Blue values come into the objective function with a negative sign. Blue would thus get assigned to hitting lowest value targets first. One could force a priority order on Blue or include specific targets in Blue's plan by expanding PB_{ijt} to PB_{ijtk}. (One could even automate the computation of these factors using the PB_{ijtk} and the E_{ik}) However, doing so requires making assumptions on the attrition rate prior to even running the model.

Attrition Rate Matrices

Both the Level 1 LP model and the Level 2 simulation require attrition rate matrices - the rates per mission at which opposing forces might be expected to destroy each other. The Level 2 simulation is used in the Stochastic Evaluators for both Level 1 and Level 2. In this section, we first derive the attrition rates for the Level 2 case, when we know the specific quantities of opposing forces. We then show how we can approximate the rates for the Level 1 LP, where precise numbers of opposing forces are not yet known. The attrition rate matrices are a function of underlying values:

$\lambda(i; i')$ = the rate per hour at which forces of type ***i*** destroy opposing weapons of type ***i'*** when engaged solely against forces of type ***i'***. (In practice, from external testing/simulations)

$\psi(i, j, t ; i', j', t')$ = the allocation of fire from weapons of type ***i***, role ***j***, attacking/defending target location ***t*** to opposing weapons of type ***i',*** role ***j'*** assigned to attack/defend at target location ***t'***, $\sum_{i'}\sum_{j'}\sum_{t'}\psi(i, j, t; i', j', t') = 1$ (computed using *fire-attribution method* below).

$z(i, j; i')$ = the number of hours per mission period spent by force type ***i*** engaging enemy weapon systems of force type ***i'***, when assigned to role ***j*** (computed via *time-over-target method* below).

These combine to form:

λ (i, j, t ; i', j', t') = λ(i; i') x ψ (i, j, t ; i', j', t') x z(i, j; i') where

λ (i, j, t ; i', j', t') = the rate/mission at which weapons of type ***i***, role ***j***, attacking/defending target location ***t,*** destroy opposing weapons of type ***i',*** role ***j',*** assigned to attack/defend at target location ***t'***.

Note that the rates are independent of the bases from which the forces originate. Base location only determines which forces may feasibly reach which targets based on the combat range (including any aerial refueling) of the forces. Further, the matrix **λ (i, j, t ; i', j', t')** will be sparse in most cases where targets are scattered over a large area; that is, there are fewer interactions among target locations.

Fire-Attribution Method

There are many schemes for computing fire-allocation as a function of weapon characteristics and weapon quantities. In our model, we assume an allocation based on the relative numbers of enemy weapon systems where:

Allocation of fire against enemy weapon system k' = $\frac{N_{k'}}{\sum_{k} N_k}$

where N_k is the number of enemy weapon systems of type ***k.*** Since these quantities can change each time period through attrition, this type of allocation must be recomputed every time period. In addition, we must determine which weapon systems should be considered in direct opposition to each other at a given time, i.e., engaged in combat, over a large battlefield.

The number of opposing weapon systems is estimated based on circular "combat zones" around target centers. The idea is that when weapon systems are assigned to defend a specific target location, they will attack offensive weapon systems within a certain radius of the target center. Conversely, weapon systems assigned to attack a specific target location will attack defensive weapon systems within a certain radius of the target center. The size of these circular combat zones are dependent on the specific force types involved and are assumed to be a function of underlying characteristics such as weapon range, performance constraints, and tactical considerations. Weapon systems assigned to attack a target location are assumed to engage in combat within these zones. The overall effect of this approach is to create "fuzzy locations" with partial interactions occurring across location boundaries. In reality, a force defending a target may cover a larger area, or it may not even have a specific target to defend. It may engage an attacking force at some distance from any final target area. However, the main purpose of

these combat zones is to allocate fire among various weapon systems that are likely to come into contact with one another.

Notional combat zones may overlap. To illustrate, suppose that an offensive weapon system is attacking target location A, as in Figure 3 below. The circle about A represents the "*target attack radius*" of the offensive force type. The circles about B and C represent the "*target defense radius*" of defensive weapon systems at those target locations. We also know the

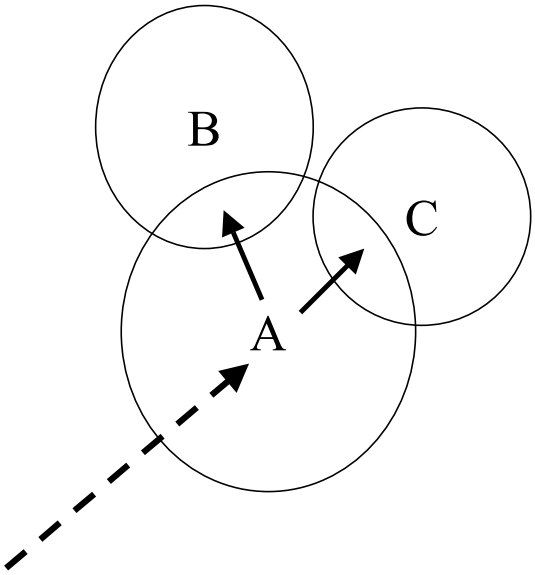


Figure 3. Allocation of Fire for Offensive Forces at Target Location A

number of weapon systems of the given type assigned to defend each target location. Assume that we want to know the number of opposing weapon systems of a given type for the purpose of allocating the fire of offensive weapon systems (as in the equation above) attacking target location A. The total number of defensive weapon systems in opposition at location A are then assumed to be those defensive weapon systems assigned to defend A as well as a fraction of those assigned to defend B and C. The fractions are determined by the size of the overlap with target location A. For example, suppose that 0.20 of target location C overlaps target location A. Then we will attribute 20% of the weapon systems assigned to defend target location C as being in opposition to offensive weapon systems attacking target location A. Thus for any given target location, the number of opposing defensive forces is estimated by adding "overlapping forces" to those specifically assigned to defend that target location. Note that each defensive force type may have its own target defense radius. Separate computations are needed to determine the attribution quantities (the "N_k") for each weapon system type. Let:

f(i, j, t; i', j', t') = the fraction of defensive forces (i', j') assigned to defend target location t' (within defense radius, r(i')) engaged by forces (i, j) assigned to attack target location t (within attack radius, r(i)). Assume f(i, j, t; i', j', t') = 1.

N(i', j', t') = the number of defensive forces (i', j') assigned to defend target location t'.

I(i,j; i', j') = 1 if weapon system (i', j') is vulnerable to weapon system (i, j); 0 otherwise

$$\psi(i, j, t; i', j', t') = \frac{f(i, j, t; i', j', t')N(i', j', t')I(i, j; i', j')}{\sum_{i'}\sum_{j'}\sum_{t'} f(i, j, t, i', j', t')N(i', j', t')I(i, j; i', j')}$$

The reverse situation is used to determine the fire allocation of defensive forces. Suppose that we are determining the allocation of fire for defensive forces of a given type at target location A as depicted in Figure 4 below. The size of the combat zones is again determined by the target defense radius of the defensive weapon systems at A, and the target attack radius of offensive forces at target locations B and C.

The defensive forces will be firing at enemy forces assigned to attack target location A, as well as portions of those assigned to attack target locations B and C.

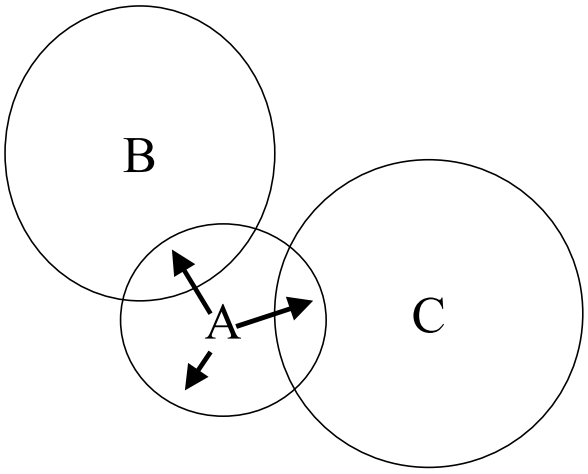


Figure 4. Allocation of Fire for Defensive Forces at Target Location A

The equation for ψ (i, j, t ; i', j', t') above holds in the defensive case with the redefinitions:

f(i, j, t; i', j', t') = the fraction of offensive forces (i', j') assigned to attack target location t' (within attack radius, r(i')) engaged by forces (i, j) assigned to defend target location t (within defense radius, r(i)). Assume f(i, j, t; i', j', t') = 1.

N(i', j', t') = the number of offensive forces (i', j') assigned to attack target location t'.

Time-Over-Target Method

The approach for time-over-target is based on the attack and defense radii discussed above. The additional parameter required is the average combat speed of the attacking forces. The conflict length is based on the speed of the attacker. We compute the average time the attacker needs to travel from the edge of the relevant combat zone to its center and back (2 radii). Let:

s(i) = the average combat speed of attacking force ***i*** . (0 or NA for fixed defenses)

Then when ***i*** is assigned an *offensive role* ***j***:

$z(i, j; i') = 2r_a(i)/s(i)$

where $r_a(i)$ is the attack radius of force ***i***. When ***i*** is assigned a *defensive role* ***j***:

$z(i, j; i') = 2r_d(i)/s(i')$

where $r_d(i)$ is the defense radius of force ***i.***

Level 1 Approximation

The formulation above for attrition rates can be adapted to the Level 1 LP model with several approximating steps. However, the inaccuracies increase the further out in time we try to project. This is one reason we always retain only the first period results from the LP in our general algorithm. Let:

$\lambda(i; i')$ = the rate per hour at which forces of type ***i*** destroy opposing weapons of type ***i′*** when engaged solely against forces of type ***i′***. (From external testing/simulations)

$\psi(i, j; i', j')$ = the allocation of fire from weapons of type ***i***, role ***j***, to opposing weapons of type ***i′,*** role ***j′***. $\sum_{i'}\sum_{j'}\psi(i, j; i', j') = 1$

z(i, j; i') = the number of hours per mission period spent by force type ***i*** role ***j***, engaging enemy weapon systems of force type ***i′.***

Which combine to form:

$$\lambda(i, j; i', j') = \lambda(i; i') \times \psi(i, j; i', j') \times z(i, j; i')$$

where λ (i, j; i', j') = the rate/mission at which weapons of type ***i***, role ***j***, destroy opposing weapons of type ***i',*** role ***j'***. The allocation of fire terms still require I(i,j; i', j') = 1 if weapon system (i', j') is vulnerable to weapon system (i, j); 0 otherwise. Now we let:

N(i,j) = the current number of weapon systems of force type ***i*** assigned to role ***j***.

Then:

$$\psi(i, j, i', j') = \frac{N(i', j')I(i, j, i', j')}{\sum_{i'}\sum_{j'} N(i', j')I(i, j, i', j')}$$

The fire allocation terms, ψ, are time dependent as they depend on the current number of forces in each assigned role/target location. We can most easily compute $\boldsymbol{\lambda}$ **(i, j; i', j', t';** $\boldsymbol{\tau}$**)** for red, where τ is the time period, since we have the parameters $PB_{ij\tau}$ (Blue plan). Let, N(i) = the total current amount of Blue force ***i***. We can compute an approximate ψ specific to each time period via:

$$\psi(i, j; i', j'; \tau) = \frac{N(i)PB_{i'j'\tau}I(i, j, i', j')}{\sum_{i'}\sum_{j'} N(i)PB_{i'j'\tau}I(i, j, i', j')}$$

Note this assumes that attrition among different Blue forces ***i***, will occur in the same proportions. It is somewhat more difficult to compute the fire allocation of Blue forces against Red - we do not know the allocation of Red forces to roles *a priori* since that is the function of the LP. Instead, we let Blue fire allocations be computed as:

$$\psi(i, j; i', j'; \tau) = \frac{N(i')I(i, j, i', j')}{\sum_{i'} N(i')I(i, j, i', j')}$$

since we *do* know the total number of Red forces of each type at the beginning of the planning horizon. (Note that we assume I(i,j; i', j') = 0 if roles j or j' are infeasible for forces i or i' respectively.) The effect is to allocate the total attack strength allocated against a Red force type into *each feasible role* within that force type. While seemingly a crude approximation, this works reasonably well because the LP tends to channel each Red force into a single optimal role in a given time period. We also must consider that due to varying rates of attrition among force types, even this approximation becomes less accurate over time. We again rely upon that fact that we only extract the first period results from each LP to compensate.

Update of AAF Target Values

AAF (anti-aircraft forces) have no intrinsic target value in the model. While they *are* mission targets, their value comes from their ability to degrade forces attacking targets that have intrinsic value such as bridges, airbases, ports, etc. This implies that within the Level 1 LP Model, AAF are treated strictly as forces. In the Level 2 simulation, they are treated as both targets and forces, but no target value is extracted from them. The single way that AAF target value enters the model is in target prioritization during the process of creating a level 2 plan for both sides (see "*Creating a Level 2 Plan*" below).

In the case of AAF targets we need to make use of Level 1 shadow prices on AAF resources and the values of the targets they are presumed to defend. The shadow price reflects the contribution to long term value. We implemented a simple pro-rating scheme as follows:

π = the period 1 shadow price on a unit of AAF resource j from the Level 1 LP (assume for now, without loss of generality, that a side has only one type of AAF).

v_t = the value of target t.

T = the set of all non-AAF targets

Recall the geographical overlap factors, f, discussed in "Attrition Matrices" above. Let:

$f_t(t')$ = fraction of non-AAF target t defended from force i by AAF at location t'

where the fraction is determined by the overlap of combat radii centered at locations t and t'. The combat radii used are the TargetDefenseRadius attribute of AAF.

$\bar{v}(t') = \frac{\sum_{t \in T} f_t(t') v_t}{\sum_{t \in T} f_t(t')}$ = the weighted average value of non-AAF targets defended by AAA at location t'

$\bar{v} = \frac{\sum_{t \in T} v_t}{|T|}$ = the average target value of non-AAF targets

If t' is an AAF target:

$$v_{t'} = \pi \frac{\bar{v}(t')}{\bar{v}}$$

The value of AAF target t' is the Level 1 shadow price of AAF scaled by the ratio of average target value defended by t' to overall average target value.

Creating a Level 2 Plan

The algorithm to build a Level 2 plan for Blue and/or Red (CreateLevel2Plan.m) assumes Red and Blue Planner input data populated as follows:

- Blue Force instances have been created. Since the algorithm uses their assigned role we assume that the role assignments in each instance are consistent with the current Level 1 Blue Plan.

- Blue Force instances have an assigned base location with known Latitude and Longitude.

- For the case of Red, one "dummy instance" of each Force type (unique name) has been created to provide force characteristics. The instance does not need an assigned role or base.

- Red Force quantity estimates are provided by Force type and Base in the form of an intelligence estimate. Each base has known Latitude and Longitude.

- The Red planner contains a populated Level1Plan (from the Level 1 algorithms) defining the Role proportions for each Force type.

- Targets on the target list of each side have been fully populated, including imputed values for AAF type targets.

The outputs of the algorithm are data structures RedForceAssignments(i,j,t) and BlueForceAssignments(i,j,t) giving the quantities assigned to each Force i, Role j, Target t. The domain of t is the Blue target list followed by the Red target list. Aircraft can be assigned to aircraft on the opposing target list in the case of the *Air Defense* role.

The algorithm can compute just a Red Level 2 plan if Blue already has an externally or previously generated Level 2 Plan. In that case, the Blue plan is held fixed. Blue can also be constrained to strike only within a specified subset of targets.

Greedy assignments generate the initial solution. A series of improving swaps of target-hitting aircraft (*Target Reduction* and *AAF Suppression*) and subsequent reallocations of *Air Defense* and *Counter Air* are used to improve the solution where each side operates in turn. The assignment processes for the two sides proceed via a "Blue Phase" and a "Red Phase". Each phase consists of three main steps in the following sequence:

1. Assign target hitting aircraft to targets
2. Opposing *Air Defense* aircraft and AAF are assigned to targets in response to (1)
3. Assign *Counter Air* to targets to escort the aircraft in (1) and defend against the aircraft in (2)

Each assignment step is based on a greedy heuristic that assigns each aircraft to the target where it can produce the highest expected value extraction (target destruction) or destruction of enemy aircraft. Aircraft can only be assigned to targets within range of their home base. Assignment to a given target can occur if a maximum allocation has not been reached, or if all targets already have a maximum allocation. In the case of target hitting aircraft, the maximum allocation is a sum of the expected target reduction proportion that will occur. The default is 2.0; this accounts for the stochastic nature of the target strikes and the fact that some aircraft may be destroyed before reaching their targets. In the case of *Air Defense* or *Counter Air*, the maximum allocation reflects the expected proportion of opposing aircraft destroyed. The default is 1.0. We provide a simplified statement of the assignment algorithm below:

1. Perform an initial greedy-based assignment of forces using a Blue Phase followed by a Red Phase. Let redTargetSet be the resulting set of targets assigned to Red target strike aircraft and blueTargetSet be the comparable set for Blue (blueTargetSet can also be specified in advance).

2. For each target, t, compute a net value to blue, V(t), using the production function V(t)=ValueEstimator(RedForceAssignments(:,:,t),BlueForceAssignments(:,:,t),t). We abbreviate this function as V(R(t),B(t),t).

3. Set:
improving = TRUE
BlueTabuSet = {}
RedTabuSet = {}
TabuTime = 10
Iteration=0;
While (improving)

Iteration=Iteration+1

3A. Blue Phase: (hold the Red Phase solution fixed.)

bestImprovement = 0;
For each pair[2] (t1,t2) of targets in blueTargetSet
For each type of target strike aircraft and target strike roles in t1
If (t1,t2,type i, role j not ε *BlueTabuSet*)

Switch one unit of target strike aircraft from target t1 to t2, providing assignment B'. Let $V_{est}(t1,t2) = V(R(t1),B'(t1),t1) + V(R(t2),B'(t2),t2)$ and V(t1,t2)= V(R(t1),B(t1),t1) + V(R(t2),B(t2),t2)

If ($V_{est}(t1,t2) > V(t1,t2)$)

[2] If blueTargetSet has > 10 targets we select a random subset of 10 targets.

Perform Steps 2 and 3 of a Blue Phase assignment to update assignment B' and to compute R'.
Compute $V_{new}(R'(t),B'(t),t)$ for each t (with $R'(t) \neq R(t)$ OR $B'(t) \neq B(t)$)

If ($\sum V_{new} - \sum V >$ bestImprovement)
bestImprovement = $\sum V_{new} - \sum V$
bestSwap = [t1, t2, i, j]
bestB = B'
bestR = R'
bestV = V_{new}

If (bestImprovement > 0)

Perform bestSwap (B = bestB, R = bestR, V = bestV)
Improving = TRUE
Reverse operation: (t2, t1, type i, role j) → *BlueTabuSet* with IncludeTime = Iteration
Else
Improving = FALSE.

Remove any elements from *BlueTabuSet* with IncludeTime < Iteration-TabuTime

3B. Red Phase: (hold the Blue Phase solution fixed)

[This is analogous to the Blue Phase except that Red seeks the swap with the largest net decrease in value to blue]

Discussion

Each greedy assignment step considers the number of aircraft available for a given role. For Blue, the individual instances define availability. For Red, the Level1Plan proportions and the estimated force quantities by base give the total numbers available. For both Red and Blue, the target location and base location are used to evaluate assignment feasibility. The target must be within the range of the aircraft.

The test, $V_{est}(t1,t2) > V(t1,t2)$, considers only the changes of value attributable to the swap of target strike aircraft. Since it does not also consider any subsequent reallocation of *Air Defense* and *Counter Air* occurring in the Blue phase it is an approximation or "hint". Its chief purpose is the speed up the algorithm, which is subject to combinatorial growth as targets increase.

The algorithm was implemented with *tabu* (Glover and Laguna, 1998) methods to avoid cycles (which were observed during testing). A cycle is where a swap keeps getting reversed, leading

to an infinite loop. Each time a swap occurs, the algorithm may not reverse the swap for a specified number of cycles.

A limit on the number of iterations can be specified in advance. A limit of 0 implies that only an initial feasible solution is desired.

The production function V(R(t),B(t),t) produces a net value to Blue from the perspective of both Red and blue, based on the Target.RedValue and Target.Value attributes. Blue operates on Blue perspective values while Red operates on Red perspective values.

The production function V(R(t),B(t),t) estimates the value extracted at each target when a given combination of Red and Blue forces are present, assuming no interactions with other nearby targets. First, attrition at the target occurs according to deterministic rate matrices. Then, target value extraction occurs according to the cumulative distribution function for the exponential distribution modeling target effects. This approach turned out to be both fast and stable. Interestingly, we found that few (less than 10) improvements could be made to the initial greedy solution for Red and Blue plans.

Stochastic Simulator

The attrition and target engagement simulation (AerialAttrition2.m) operates at planning Level 2. Since Level 2 incorporates target and mission planning, the simulation model must distinguish between target locations. The input includes the number of aircraft by side, force type, and role assigned to attack or defend each target. These quantities were determined using the algorithm described in *Creating a Level 2 Plan* above.

The simulation consists of three main phases:

Phase 1 Pre-Target: *Counter Air*, *Target Reduction*, and *AAF Suppression* on each side fights it out with opposing AAF and other *Air Defense* for 1/2 time period. Attrition reduces the force quantities.

Phase 2 Target: Surviving *Target Reduction* and *AAF Suppression* aircraft on each side engage Targets to determine a score. The capability indices at the targets are reduced accordingly.

Phase 3 Post-Target: Aerial battle of Phase 1 recommences for remaining 1/2 of time period with remaining aircraft

The planning logic assumes that the primary objective of the air campaign is to support the forces (hit Targets) on the ground. If more planes can be allocated to this role, the score will be higher. Clearly, a myopic strategy of assigning too many aircraft, too soon to *Target Reduction* may not work well. The enemy *Air Defense* may need to be eroded first so that *Target Reduction* forces do not all get destroyed.

Attrition Logic

The attrition portions of the simulation (Phases 1 and 3) are designed as a multi-weapon type, stochastic "Lanchester" model. The state of each side (Red/Blue) is provided by a state-vector of weapon-system types (as in Popken and Cox, 2000) by location. Each unit on each side has a set of parameters designating the rate at which they can destroy opposing force types. To illustrate:

λ_{ij} = the rate (per planning period) at which forces of type ***i*** destroy opposing forces of type ***j*** when engaged solely against weapons of type ***j*** (assume that i and j incorporate the notion of force type, role, and location).

Since each unit can potentially destroy more than one opposing type, a probabilistic fire allocation scheme has been devised. It is based on the "fractional allocation method" discussed in Anderson and Miercort (1989). The probability of a unit of weapon type ***i*** on side ***s***, selecting an opposing weapon type j on side ***s'*** is given as:

$$A_{ij}^{s} = \frac{C_{ij}^{s} W_{j}^{s'}}{\sum_{j=1}^{N^{s'}} C_{ij}^{s} W_{j}^{s'}}$$

where:

A_{ij}^{s} = the allocation of fire from a weapon of type *i* on side *s* when that weapon is engaging an enemy of type *j*; i = 1,2,3, .. N^{s} , j = 1,2,3, .. $N^{s'}$, s $\in$ {R, B}. Note that $\sum_{j=1}^{N^{s'}} A_{ij} = 1$.

C_{ij}^{s} = the corresponding fire allocation weighting coefficient

W_{i}^{s} = the number of weapons of type i on side s; i = 1,2,3, .. N^{s}, s $\in$ {R, B}

N^{s} = the number of different weapon types on side s; s $\in$ {R, B}

Note that this allocation rule is state-dependent in that it depends on the current number of surviving weapons of each type on each side. A_{ij}^{s} is constructed to concentrate fire on the weapon types with greater number. In some cases, one can improve the allocation by appropriate selection of C_{ij}^{s}. Anderson and Miercort (1989) suggest that:

$$C_{ij}^{s} = E_{i}^{s} P_{ij}^{s} E_{j}^{s'} P_{ji}^{s'}$$

where:

E_i^s = the average number of engagements per time period made by a weapon of type *i* on side *s* (against all enemy weapons). Note that E_i^s is the inverse of the mean of the interfiring time distribution for weapon system *i*; i = 1, 2, 3, .. N^s, s ∈ {R, B}.

P_{ij}^s = the probability of kill per engagement by a weapon of type *i* on side *s* when that weapon is engaging an enemy of type *j*; i = 1, 2, 3, .. N^s , j = 1, 2, 3, .. $N^{s'}$, s ∈ {R, B}.

This formulation tends to focus fire on targets that are most effective by weapons which are the most effective against them. However, problems arise when opposing forces are relatively defenseless but are nevertheless very desirable for targeting (e.g. Mig-29 vs. B52). The equation above would give them a low or zero allocation. For this reason, we use a simple allocation where the parameters, C_{ij}^s, are set to 1.0; allocation is based on relative numbers of opposing forces at a given location.

Next, we need to know how to determine what gets destroyed and when in the simulation. We define the "total destruction rate" as:

λ_j^s = the total rate at which weapons of type ***j*** on side ***s*** are being destroyed by weapons on the opposing side, or

$$\lambda_j^s = \sum_{i=1}^{N^{s'}} A_{ij}^{s'} \lambda_{ij}^{s'} W_i^{s'}$$

We assume that the casualty process is a Markov process with rate, λ_j^s. Therefore the time between kills for each weapon type ***j*** on side ***s*** has an exponential distribution with mean $(1/\lambda_j^s)$ (similar to the Bonder-Ferrell approach to determining Lanchester rate coefficients (Taylor, 1983)). But since these rates change with the force populations, they must be updated after each kill. This process occurs within the following attrition simulation algorithm:

Set W_i^s = the surviving number of weapons of type i on side s; i = 1,2,3, .. N^s, s ∈ {R, B}.
clock=0

while clock < T

1. compute the fire allocation fractions A_{ij}^s
2. compute the total destruction rate for weapons of type ***j*** on side ***s***: $\lambda_j^s = \sum_{i=1}^{N^{s'}} A_{ij}^{s'} \lambda_{ij}^{s'} W_i^{s'}$
3. determine the next time to destruction, t_j^s, for each weapon type by drawing from an exponential probability distribution $E(\lambda_j^s)$

4. find $t_{min} = \min_{j,s}(t_j^s)$

5. if (clock + t_{min} < T)

    a. for s and j such that $t_j^s = \min_{j,s}(t_j^s)$, set $W_j^s = W_j^s - 1$

6. set clock = clock + t_{min}

Target Engagement Model

Target engagement occurs during the second or middle phase of the simulation (via EngageTargetList2.m in the accompanying MATLAB files). The surviving *Target Reduction* and *AAF Suppression* aircraft on each side hit the target in the target location to which they are assigned.

For each target location, the model works through the targeting aircraft on each side. First, the current time period must be within the window of opportunity defined for the target. The actual damage done by an aircraft is then determined probabilistically. The damage is modeled as an exponential random variable with a mean value equal to the target effectiveness (average capability reduction by target) of the aircraft. After each strike, the target lists are updated and value reduction is accumulated to score the engagement. No value can be extracted from a target once its capability falls below a desired damage level.

Note that each target has two different "values", one according to Blue's value system, the second according to Blue's perception of Red's value system. The target engagement model returns the scores from both value systems. (Higher level algorithms will assume that each side seeks to optimize decisions according to their own value system.)

Results and Discussion

Database

The data used to populate the domain model came primarily from two sources. The first is a spreadsheet dataset used by AFRL/IFSB for model testing known as "the Korean Scenario". It provides data for a subset of US forces - on the order of several hundred individual aircraft - in and around South Korea. The second major source, a well known public web site (http://www.globalsecurity.org), was used to obtain data regarding North Korean forces and bases. The test database contains 13 Blue force types and 7 Red (North Korean) force types, including both aircraft and AAF on each side. Blue has 15 base locations, each of which is a target for Red. Red has 201 targets, including bases, SAM sites, ground force concentrations, and various infrastructure targets.

Some of the domain attributes were approximated with plausible, if not completely accurate, values. Approximate data suffice to explore the dynamics of the planning algorithms. The approximations included the combat effectiveness (kills/hr when a unit of force type ***i*** is in combat against force type ***j***), target effectiveness (capability degradation per mission by force ***i*** against target ***t***) and the target values. To compute combat effectiveness, we used some simple heuristic rules based on a relative force strength index. For target effectiveness, we used a scale that was based on the lbs of munitions carried by an aircraft versus an estimate of lbs. of munitions needed to destroy a target. For target values, we used estimates based on examples from the AFRL data set. (These rules are documented in the GenerateTables.m file in the TestingAndMaintenance folder of the accompanying MATLAB files. Also see the installation instructions on how to use this function file to generate revised data).

Verification Testing

The purpose of this testing is to ensure that no errors or warnings occur over a range of input data values provided by a random data generator and by the final Korean scenario data set. All algorithm outputs appeared properly formed with reasonable values. All controls on the MATLAB GUI panels (see Appendix A: AFSimPlan Software) functioned as expected with correct displays and invoked behaviors.

Performance Testing

The purpose of this testing was to determine the runtime requirements of the major algorithms. We first used the MATLAB "profile" function to find and eliminate bottlenecks. The next step was to benchmark the test machine using the MATLAB "bench" function (Figure 5). This provides a baseline for comparing the performance results to those obtained on a different machine.

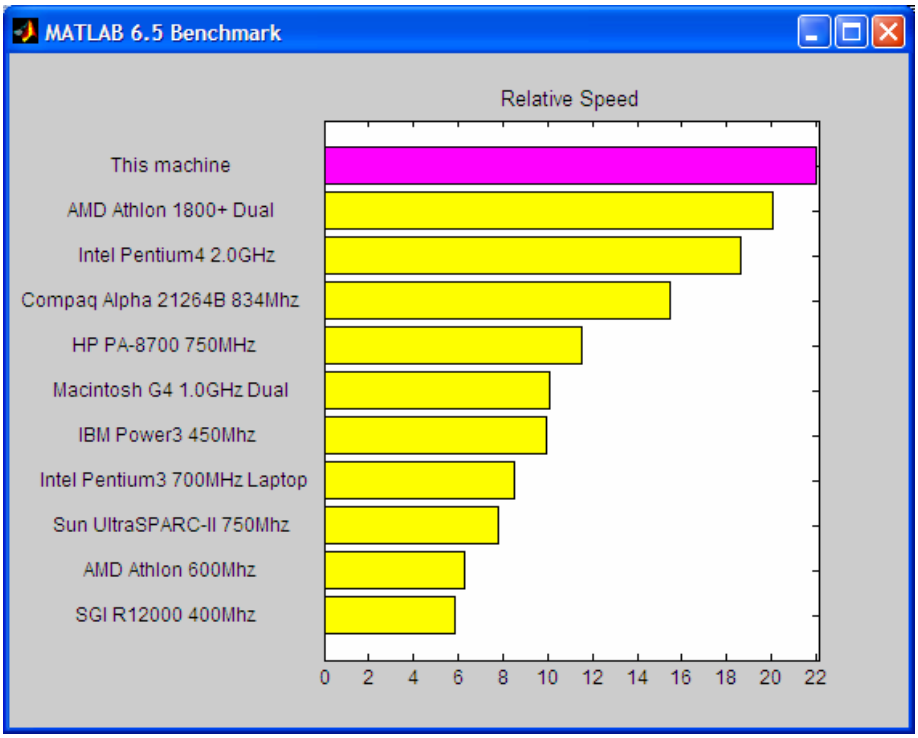


Figure 5. MATLAB Benchmark for the Test Machine

Level 1

The following performance tests were performed:

1. Record the average time required to perform a stochastic evaluation of a given Blue Level 1 plan with 10 sample iterations per evaluation.
2. Test the plan optimization time for a 5 period plan:
    a. Vary N (the max # of improvement steps used to obtain a revised Blue Plan) → set of N = {100,200,…. Max}
    b. Set the Blue plan to an uninformed (flat) prior distribution for the feasible roles for each force. Set K = 10 = number sample evaluations per Level 1 Blue plan. Set the horizon weight to 1.0.
    c. Record the total time required to complete optimization for each optimization level, N.

N (improvement steps)	Total time (hr)	Cumulative Time/Step (min)	Time/Step for Interval
100	7.26	4.36	4.36
200	13.31	3.99	3.63
300	18.86	3.77	3.33
400	23.98	3.60	3.07
500	28.89	3.47	2.95
600	33.79	3.38	2.94
700	38.63	3.31	2.91
726	39.85	3.29	2.81

Table 1. Level 1 Performance Testing Summary

We see that each plan evaluation requires several minutes for 10 sample iterations, an average of 4.36 minutes each for the first 100 improvements and 2.81 for the last 26. The evaluation times become smaller as the forces are concentrated into fewer roles than the initial flat prior distribution. The times are also somewhat less for smaller horizon weights (not shown), as forces are further concentrated into fewer roles. We also see that the time required to fully optimize the Blue Plan can be very lengthy. In the case of horizon weight = 1, almost 40 hours were required on the benchmark machine. It should be noted that less time is required for smaller horizon weights; however, this is not due to efficiency. With lower horizon weights the relative marginal values (shadow prices) are smaller, causing the plan improvement algorithm to quit because of a lack of statistically significant differences in role allocations. For example, at horizon weight = 0, the algorithm quits after about 350 improvement steps. At that point it has operated on allocations in periods 1, 2, and part of 3. It cannot detect differences in allocations beyond period 3. At horizon weight = 1, all forces, roles, and periods were operated upon.

Level 2

The following performance tests were performed:

1. Set $\boldsymbol{K}$ = 100 = the number of sample iterations per Level 2 Blue plan. Set the Level 1 Blue plan to an uninformed (flat) prior distribution for the feasible roles for each force.
2. Vary the number of improving cycles, N, in the Level 2 Optimizer→N = {0,10,20,30, etc}. Record the total time required to *create* a Level 2 Plan for the set of N.
3. Record the time required to *evaluate* each given plan with $\boldsymbol{K}$ =10 sample iterations.

At Level 2, we first create a Level 2 Blue plan using the greedy heuristic described previously. We then evaluate the plan with the Level 2 Stochastic Evaluator. In the tests reported below, the random number generator was reset before each "Create" and each "Evaluate" so that the results could be fairly compared.

N	Create Time (sec)	Evaluation Time (sec)	Mean Plan Value	Std. Dev
0	3.275	408.13	6458.29	206.35
10	51.91	345.58	6465.50	154.60
20	53.30	349.90	6465.50	154.60

Table 2. Level 2 Performance and Face Validity Testing Summary

We see that the plan creation time is relatively quick (a few seconds) if no improving iterations are made. Adding N=10 improving iterations provides a big jump in run-time to almost a minute. However, there is no corresponding increase in Evaluation Time. The results above actually show a small decrease but that is merely random variation. The implications of the Plan Value as a function of N are discussed in the next subsection below.

Face Validity

Level 1

It is important for plans to reflect varying emphasis on short versus long term considerations. In the short run, there is no value given to survivability beyond the planning horizon. In this test we create Level 1 Blue plans using different horizon weights.

1. Vary the horizon weight to place varying emphasis on near versus long term – horizon weight = {0, .5, 1.0}. Set K=10. Set the planning horizon to 5 periods.
2. For each horizon weight
   d. Initialize the Blue Level 1 plan to an uninformed (flat) prior.
   e. Create a fully re-optimized Level 1 Blue plan and record the plan result.

horizon weight	Counter Air	Air Defense	Target Reduction	AAF Suppression	Other
0	2.30	4.90	47.47	4.07	6.27
0.5	0.70	19.90	41.67	1.00	1.73
1	1.20	23.00	39.60	0.20	1.00

Table 3. Total Allocations By Role Over the Planning Horizon

Table 3 summarizes the resulting plans by summing role allocations over all forces and time periods. As expected, *Target Reduction* is deemphasized with higher horizon weights. The other significant change is in *Air Defense*. At the same time, the other three role allocations show decreases. That *Air Defense* increases, rather than say, *Counter Air*, reflects the objectives in our model. Both Red and Blue are trying to maximize target value extraction. By assigning aircraft to *Air Defense*, Blue can both protect targets and reduce Red forces through attrition. This improves longer term survivability since Red then has fewer planes with which to attack Blue forces.

When the horizon weight = 0, operating the plan optimization algorithm would again be expected to increase the discounted net Blue plan value. For higher horizon weights, the optimizing algorithm should produce less net Blue value over the planning horizon, since survivability becomes a greater consideration. Figures 6 and 7 show the discounted net Blue plan value as a function of improving steps for the cases with horizon weight = 0 and 0.50.

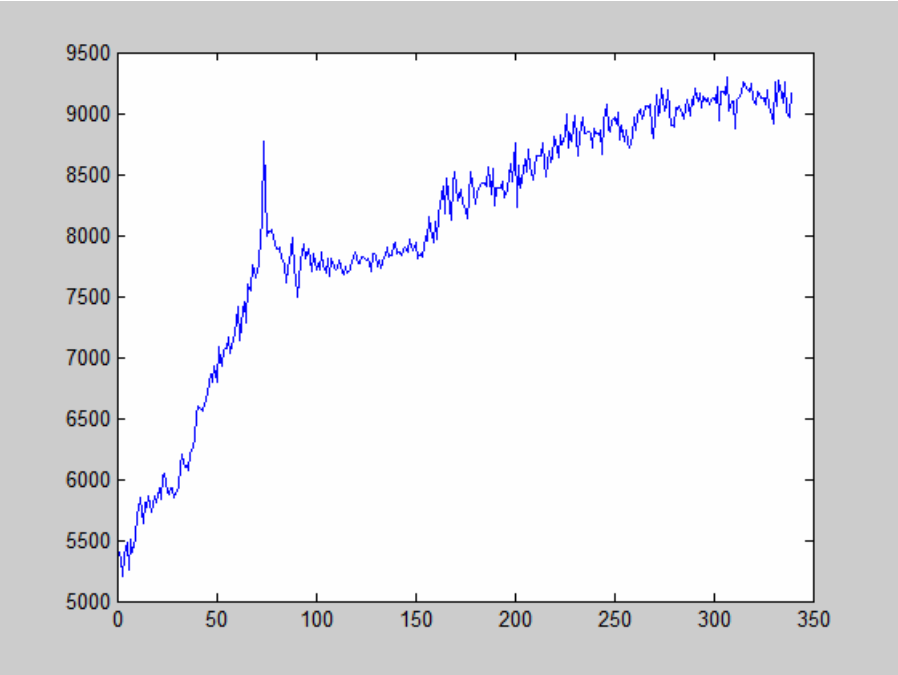


Figure 6. Net Blue Plan Value vs. Improving Iterations (Horizon Weight=0)

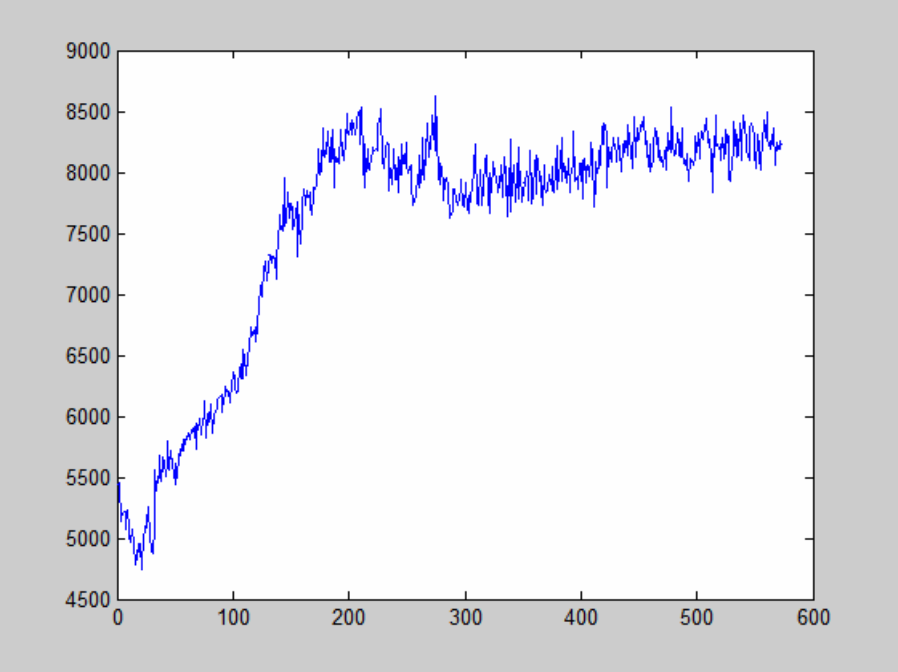


Figure 7. Net Blue Plan Value vs. Improving Iterations (Horizon Weight = .50)

Both cases show a generally steady upward trend in net Blue plan value. With a horizon weight of zero, the net time discounted value extracted by the final plan is approximately 65% higher than for the original plan. With a horizon weight of 0.50, the value increases by approximately 50%. However, both cases display regions where the net Blue plan value has a locally decreasing trend, most markedly when the horizon weight = 0, between 75 and 100 improving iterations. This is not random variation: the phenomenon was observed for different random number seeds. We investigated the reallocations occurring at that point and found that forces were being shifted from *Counter Air* to *Target Reduction.* Our current hypothesis on the source of these dynamics is that they are caused by elements of the current linear programming formulation, which assume that Blue can always extract target value at an average rate per aircraft type. Red, on the other hand, has an explicit upper bound on the maximum target value extraction. A similar constraint on Blue could render a Blue input allocation infeasible, thereby requiring performance reducing workarounds. So in some situations, Blue *Target Reduction* forces will "overrun" red targets – there are not enough Red targets to hit. This over-allocation is then reflected in the simulation. The planes moved from *Counter Air* to *Target Reduction* by the LP subject all Blue forces to greater attrition. At the same time, the former *Counter Air* forces have no targets remaining to hit. Therefore the overall net target extraction is reduced until additional adjustments are made by the optimizer. Optimizing the plan for a relatively few steps from a given starting point is not guaranteed to increase value when Red targets are near exhaustion.

Level 2

The Level 2 validation test examined how the net Blue plan values change as a function of the number of improving iterations using the Level 2 plan creation algorithm. Of course, improvements are being made simultaneously to both Red and Blue plans; that is, each step involves first a Blue improvement, and then a Red improvement. After several improving iterations, it is possible to wind up with a net Blue plan value with little net change. Table 2 above shows the plan values as a function of the number of iterations. Recall that the algorithm begins with a greedy based initial assignment for each side, and then the improving iterations operate on the initial plan. The improvements made little difference (and no incremental improvements after 10 iterations in any test case examined), suggesting that the initial plan may suffice in many cases, unless a more effective improvement routine can be discovered.

Statistical Comparisons between Planned and "Actual" Conflict

The last set of tests exercise the algorithms as they might be employed during actual use, applying them to the same simulated conflict used by the internal planning algorithms. (This highlighted differences due to approximations made by the planner rather than differences due to mismatches between the simulation and reality.) Figure 8 illustrates the evaluation process.

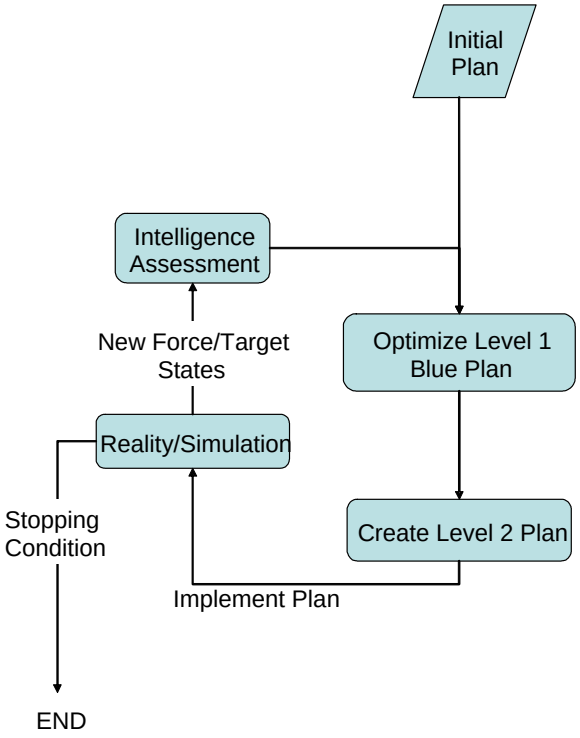


Figure 8. Model Use during a Planning Cycle

The first test demonstrates how the planner adds value to the plans during the operational planning cycles. The stopping condition for the conflict is that Blue can no longer extract net positive value, (e.g., because all Red targets are destroyed). With the horizon weight set to 0 and the discount factor to 0.80, the optimizer was allowed to make N improving iterations, where N ε {0, 50, 100}. The initial plan is the uninformed (flat) prior, where forces are allocated evenly across feasible roles for five periods We assumed perfect intelligence estimates for this particular test (relaxed in the next test), so the number of samples per evaluation is K= 1. After each cycle, the plan is updated by discarding the already used first period. Then, if there are sufficient periods left in the conflict to allow it, we add on a new period to the plan, also set to an uninformed prior. It could take much iteration to fully optimize a five-period plan. However, we are only using the first period of the plan at a time, so a full optimization is not necessary. In practice, a longer plan would be constructed to facilitate integrating air warfare plans with other military planning outside of the scope of this model. Five runs were made for each value of N, with results shown in Table 4 below.

N/Cycle	**Stopping Point *t***	**Mean Value Extracted**	**Std. Dev.**
0	10-15	7002.8	133.70
50	5-6	8884.7	283.54
100	3	9381.3	346.19

Table 4. Value Extraction during a Conflict

Clearly, the plan optimization adds significant additional value from 0 iterations to 50 iterations. The difference in mean value extracted at 100 versus 50 iterations is also statistically significant but shows that there are decreasing returns from the additional iterations. (However, the additional iterations would also help build the plan for future periods, which could be important for coordinating plans with external units.)

The second test quantifies the performance of the planner with respect to the accuracy of the intelligence. The previous test assumed that the "Intelligence Estimator" of Figure 6 above was "omniscient" – it always knew the exact number of Red forces remaining. We now use a "poor", systematically biased, Intelligence Estimator. It is given an "assumed detection probability" of 0.90. That is, Blue believes that it can detect 90% of Red's forces, and scales the result accordingly. However, the "true detection probability" is actually 50%. Blue will consistently underestimate Red forces. Both scenarios involve the same number of "actual" Red forces. Setting the horizon weight to 0.50 as before, the number of samples per evaluation, K, will be 10 with poor intelligence and 1 with perfect intelligence. Table 5 summarizes the results from five runs of each scenario.

Scenario	Mean Value	Std. Dev.
Omniscient (Perfect Intel)	7572.6	112.81
Poor Intel	8094.5	104.25

Table 5. Discounted Net Blue Values Extracted with Varying Intelligence Quality

The set of runs with poor intelligence extracted significantly higher Blue value than those with perfect intelligence! This is because, with a horizon weight of .50, we are optimizing a weighted average of Blue value and survivability. Refer back to Figures 6 and 7. The optimized net Blue value is lower at a horizon weight of 0.50 versus a horizon weight of 0. Poor intelligence thus creates a situation similar to having a lower horizon weight. When intelligence is poor (Red is underestimated), the optimizer believes that survivability considerations are less necessary, since there are seemingly fewer Red forces to contend with. The resulting plan overemphasizes Target Reduction, with a higher than anticipated attrition rate.

Conclusions

The project successfully demonstrated the use of an integrated planning hierarchy as a framework for air warfare planning. It showed that linearization of the decision space can be used to approximate a problem that would otherwise be intractable due to its size and uncertain dynamics. Within the evaluator, linear programming and simulation operated in tandem over an extended planning horizon to generate optimized Red responses, assumed outcomes, and relative marginal force values. Uncertainty was successfully handled with sampling approaches. We showed how more detailed plans can be automatically generated that were consistent with top level plans. Last, we demonstrated how the plan optimization algorithms can be embedded in an operational planning cycle operating over a multi-period conflict, and showed that the quantitative impacts of poor intelligence and false beliefs about enemy forces can lead to inappropriate plans. This provides a quantitative foundation for assessing and managing Blue's risks by selecting plans that are robust to uncertainties about intelligence, while perhaps reducing performance on the "best guess" intelligence (in case it proves to be wrong.) Optimally hedging bets (i.e., current decisions) against imperfect intelligence is an important topic for future development that can be explored starting from the approach illustrated in Table 5.

With several minutes required to evaluate each plan, fully optimizing a plan over a 5-period planning horizon required many hours on the benchmark machine. There may be ways to address this issue through more efficient solution of the nonlinear mathematical programs at the heart of the planning system and/or by using statistical approximations to multi-period value functions (Lagoudakis and Parr, 2002). Our approach involved successive solution of linear programs, requiring $(T+1)(T+2)/2 - 1$ linear programs for *each* sample of the Level 1 Stochastic Evaluator, where T is the number of periods in the planning horizon. The advantage of the linear programs is their automatic computation of relative marginal resource (force) values in the form of shadow prices.

Regarding potential future research, the system could be enhanced by making the target values and capabilities dynamic. The current model is driven by the "extraction" of target value. Dynamic target values could be generated by a monitoring program that oversees all elements of a conflict and computes current relative values consistent with an Effects Based Operations (EBO) approach. It could build on the current capability of the model to represent differing Blue and Red perceptions of target value. It would also be reasonable to express target values as probability distributions, given the typical uncertainties associated with targeting and damage assessments. Further, target capabilities could change to reflect repair of damaged targets. Lastly, the results of this project also suggest that research into extending the planning process to the third level (Routes/Engagements) is warranted. Many of the basic elements of the approach - linearization, sampling, high-level simulations, and rolling forward through a planning horizon - would likely be very useful in such an extension.

References

Anderson, LB and FA Miercort, Combat: A Computer Program to Investigate Aimed Fire Attrition Equations, Allocation of Fire, and the Calculations of Weapons Scores, IDA Paper P-2248 (DTIC AD-A213660), 1989.

Bertsekas DP, Tsitsiklis JN, Wu C. Rollout Algorithms for Combinatorial Optimization, *Journal of Heuristics*, v.3 n.3, p.245-262, November 1997

Booch G, J Rumbaugh, and I Jacobson. The *Unified Modeling Language User Guide*. Addison-Wesley, Boston. 1999.

Carson Y and A Maria. Simulation optimization: Methods and applications. *Proceedings of the 1997 Winter Simulation Conference* ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson 1997.

Cave, WC and TE Busch. Theoretical underpinnings of predictive battlespace awareness, In *Enabling Technologies for Simulation Science VII*, Alex F Sisti and Dawn A Trevisani, (eds.), Proceedings of SPIE Vol. 5091, 17-26, 2003.

Cruz, JB, MA Simaan, A Gacic, and L Yong. Moving Horizon Nash Strategies for a Military Air Operation, *IEEE Transactions on Aerospace and Electronic Systems*, **38(3)**, 989-999, 2002.

Dippon J. Accelerated randomized stochastic optimization. *Ann. Statist*., 31(4), 1260-1281, 2003**.**

Dresher, M. *The Mathematics of Games of Strategy: Theory and Applications*. Dover Press, 1981.

Dreyfus, HL. *What Computers Still Can't Do*. MIT Press, Cambridge, MA, 1994.

Glover, F and M Laguna. *Tabu Search*. Kluwer, Boston. 1998.

Heise, SA and HS Morse. The DARPA JFACC program: modeling and control of military operations, In *Proceedings of the 39th IEEE Conference on Decision and Control*, Vol. 3, 2551-2555, Sydney, Australia, 12-15 Dec 2000.

Isaacs, R. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. John Wiley & Sons. New York. 1965.

Karlin, S. *Mathematical Methods and Theory in Games, Programming, and Economics*, Dover Press, Mineola, NY, 1959.

Lagoudakis MG and R Parr. Value Function Approximation in Zero-Sum Markov Games. *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*. Adnan Darwiche and Nir Friedman (eds.), Alberta, Canada, Aug 1-4 2002.

McEneaney, WM and K Ito. Stochastic Games and Inverse Lyapunov Methods in Air Operations, In *Proceedings of the 39th IEEE Conference on Decision and Control*, Vol. 3, 2568-2573, Sydney, Australia, 12-15 Dec 2000.

McEneaney, WM, BG Fitzpatrick and IG Lauko. Stochastic Game Approach to Air Operations. Paper Submitted to *IEEE Trans. Aero. Elec. Systems*, 2003. (Currently available at http://math.ucsd.edu/~wmcenean/pubs/jfaccpaper.pdf)

Mukai, HA, Tanikawa, I. Tunay, A. Ozcan, I. N. Katz, H. Schättler, P. Rinaldi, G. J. Wang, L. Yang and Y. Sawada. Game Theoretic Linear-Quadratic Method for Air Mission Control, In *Proceedings of the 39th IEEE Conference on Decision and Control*, Vol. 3, 2574-2580, Sydney, Australia, 12-15, Dec 2000.

Popken, D and LA Cox. An Investigation of System Identification Techniques for Simulation Model Abstraction, Technical Report AFRL-IF-RS-TR-2000-8, Air Force Research Laboratory, Information Directorate, Rome, NY. 2000.

Taylor, JG. *Lanchester Models of Warfare, Volume II*, Military Application Section of the Operations Research Society of America, Alexandria, VA, 1983.

Yost, KA and AR Washburn. The LP/POMPD Marriage: Optimization with Imperfect Information, *Naval Research Logistics Quarterly*, **47**, 607-619, 2000.

Appendix A: AFSimPlan Software

Overview

The AFSimPlan software (see Figure A1) implements the algorithms described in this report. The algorithms are written in MATLAB, and are accessed from the MATLAB command window via MATLAB GUI controls as described in the Users Guide below. Since the MATLAB files are essentially source code, the user can also make revisions to the algorithms or default parameters with the MATLAB editor. The MATLAB files reside in the C:/Program Files/Systems View/AFSimPlan/MATLAB directory.

The working data file, planningData.mat, is a MATLAB format compressed data file that contains the MATLAB objects (see Appendix B: Domain Model) needed to run the algorithms. It resides in the C:/Program Files/Systems View/AFSimPlan/data directory. MATLAB Load and Save commands embedded within the application operate on the data file.

The application framework also contains an SQL database (in the same data folder), containing the same type of data as planningData.mat, but in a relational database format. The SQL database provides longer term storage of domain data. It also provides a standardized format more accessible to other external data systems.

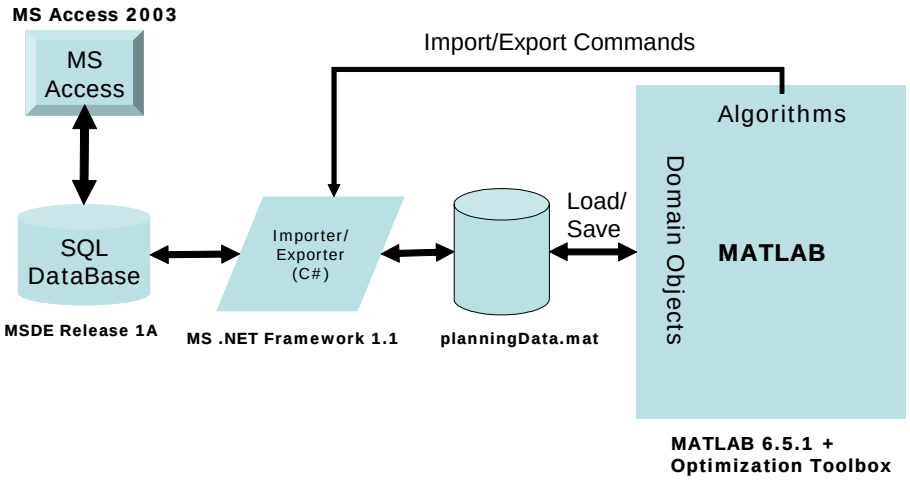


Figure A1. AFSimPlan Architecture

The user can view and make edits to the SQL data tables through an interface set up in MS Access. The user needs to navigate to the C:/Program Files/Systems View/AFSimPlan/data/OpSim-Phase1_SQL.mbd file and open it with MS Access. The first time the user does this he needs to refresh the links between tables (see Installation Instruction below).

The user will need a software tool to “Attach” and “Detach” the SQL database. The database must be detached before it can be moved or copied. For convenience, the install will provide a copy of the freeware, “MSDE Manager®”. The user could also use other tools or even SQL Server to perform the same function.

The AFSimPlan application also provides a means to import/export between the working file, planningData.mat, and the SQL files. The application modules that perform this task were written in C# and are implemented in the MS .NET framework. These modules operate behind the scenes and are invoked from the MATLAB based GUI controls.

User Guide

Level 1 Analysis

This portion of the application is invoked by typing "level1analysis" at the MATLAB command prompt. The control panel shown in Figure A2 below will be displayed.

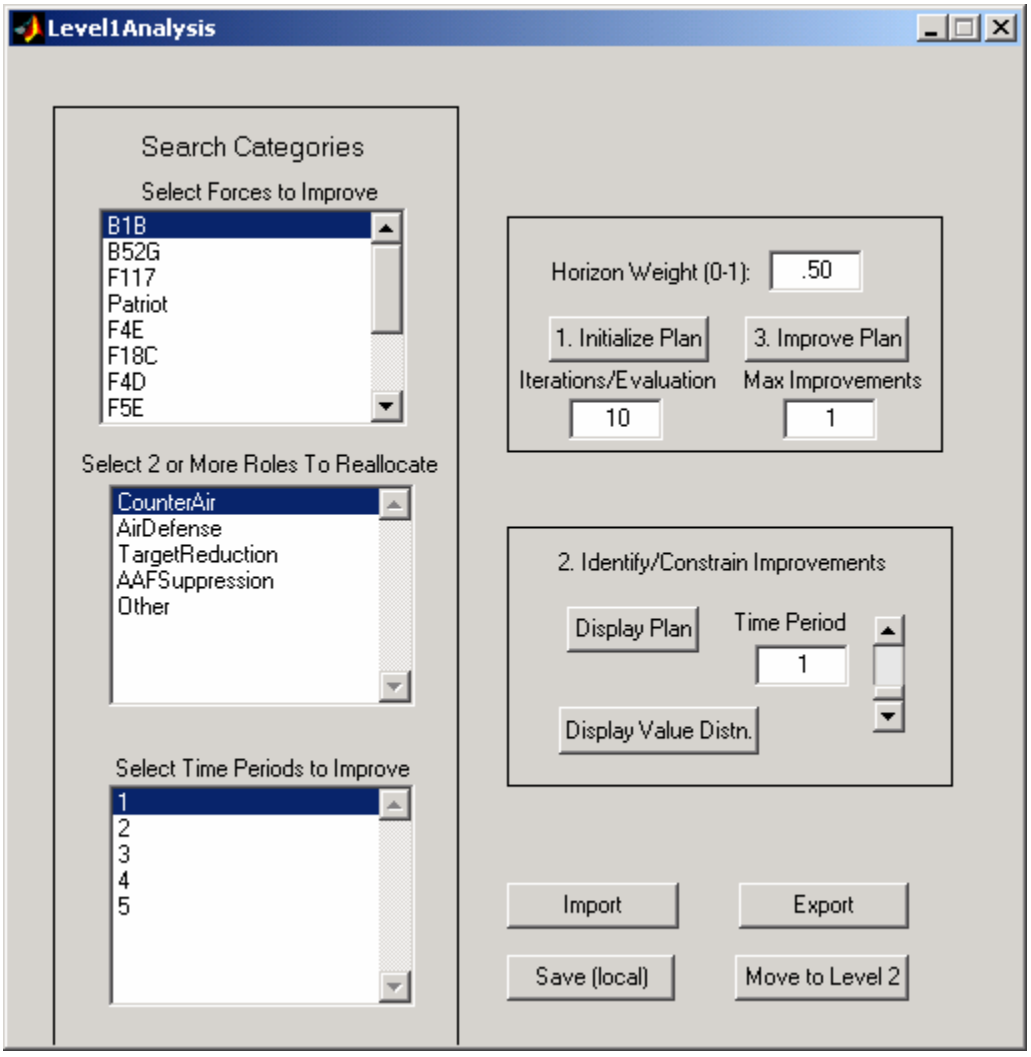


Figure A2. Level 1 Controls

1. Initialize Plan (Button). This button invokes the Stochastic Evaluator, providing a set of sample net Blue values for the current Level 1 Blue plan (role allocation). The function makes use of both *Iterations/Evaluation* and *Horizon Weight* (approx. 4.5 minutes on benchmark machine for 10 iterations).

Iterations/Evaluation (Edit Field). The number of samples desired for each evaluation of the current plan. Note that each sample requires about 30 seconds on the benchmark machine.
Horizon Weight (Edit Field). The relative emphasis placed by the plan on short term versus long term considerations. A (minimum) value of 0 implies short term, while a value of 1.0 (maximum) implies long term. A short term perspective places no value on aircraft surviving beyond the planning horizon.

2. Identify/Constrain Improvements:

Display Plan (Button). Displays the results of the last completed plan evaluation (see Figure A3 below). Two types of information are displayed for each Blue Force Type. The first, Marginal Value, is the average resource price for that force returned by the linear programs within the Stochastic Evaluator. The Marginal Value reflects the expected contribution to the net Blue value obtained by adding an additional unit of that force. Since the values are specific to roles, the height of the colored band for a role corresponds to the relative Marginal Value for a given force/role combination. The second type of information is the Current Allocation. It shows the proportional allocation to each force/role allocation.

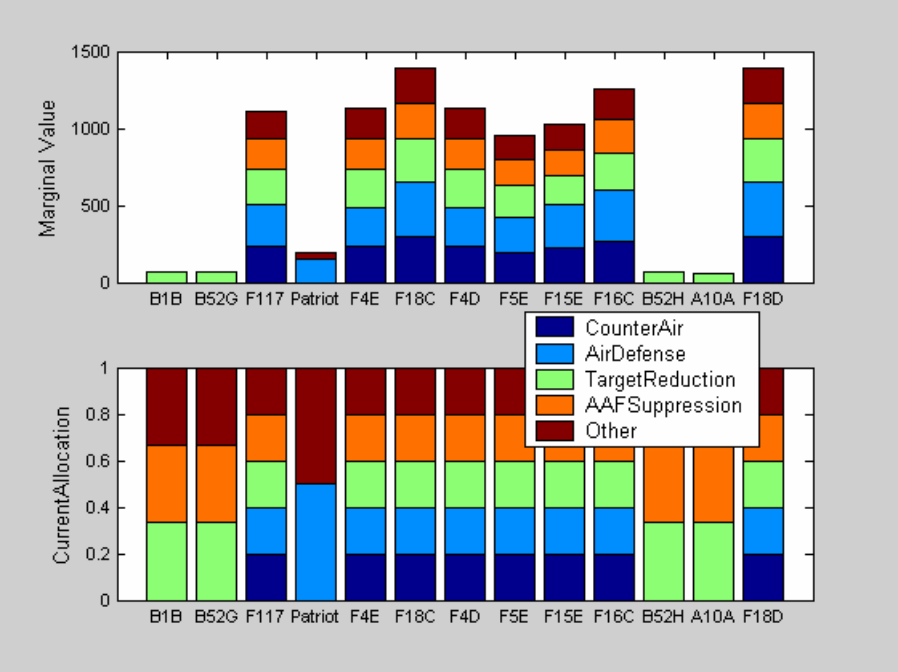


Figure A3. Sample Plan Display

Improvements for a given force type can be visually identified by scanning for large differences in the height of the role contributions to Marginal Values. If allowed by the

Current Allocation, a swap of forces between the roles would lead to an increase in net Blue value. For example, Blue would be better off if F18C's were moved from *AAF Suppression* to *Air Defense*. The Current Allocation shows that this swap would be feasible.

Time Period (Edit Field). This function makes use of the *Time Period* field to determine what time period of the plan should be displayed.

Display Value Distribution (Button). This displays the results of the most recent plan evaluation with respect to the net Blue value obtained (see Figure A4 below). The graph is the empirical cumulative probability distribution for net Blue value (the probability that the value is less than or equal to the x coordinate).

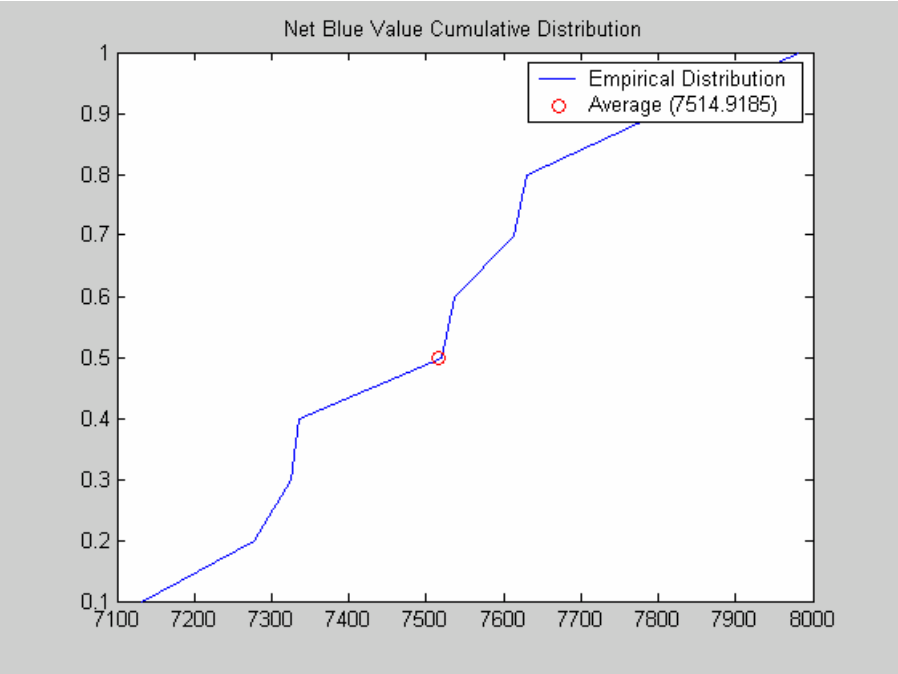


Figure A4. Sample Net Blue Value Cumulative Distribution

Search Categories

The user has the option of constraining any search for improvements to the current plan. Only the Forces, Roles, and Time Periods selected from the list boxes on the left will be searched for improving reallocations. The user can use SHIFT-left mouse button to select groups of items in each list box, and CTRL-left mouse button to select multiple non-consecutive items.

3. Improve Plan (Button). This button invokes a "hill climbing" algorithm that finds improvements to the current Level 1 Blue plan. The algorithm searches over the Search Categories selected above. If the user makes no selections. the search will be unconstrained. The search algorithm uses the Stochastic Evaluator to analyze each (re)allocation, thus it utilizes

both *Iterations/Evaluation* and *Horizon Weight*. The algorithm completes when no further statistically significant improvements can be found, or once it makes *Max Improvements* number of improving reallocations. Since the search process can be lengthy, it is often advisable to limit the total iterations via *Max Improvements*. During the search, a display will appear (after the initial evaluation) showing the current net Blue value as a function of the number of improvements as in Figure 6 of the Testing section. After every improving step, the computer will make a "double-beep" sound and the net value graph will update (approximately 4.5 minutes per improvement on the benchmark machine for 10 iterations/improvement, less for large numbers of improvements – see Testing section).

> *Max Improvements* (Edit Field) The maximum number of improvements allowed by the Improve Plan function.

Import (Button). This will first invoke the import/export utility. The utility will update the local copy of the data (planningData.mat) from the SQL database. The local copy will then be read into MATLAB. Last, the control panel will be refreshed with the new data. A second MATLAB command window (minimized) will briefly appear during this process.

Export (Button). First the current data is saved to the local copy of the data (planningData.mat). Next, the import/export utility will update the SQL database with the output values from the current analysis (it does not update the basic domain data). A second MATLAB command window (minimized) will briefly appear during this process.

Save (Local) (Button). This saves the current data in MATLAB to the local copy of the data (planningData.mat).

Move to Level 2 (Button). Invokes the control panel for Level 2 Analysis

While using the Stochastic Evaluator (*Initialize Plan* or *Improve Plan*), the following warning may appear in the MATLAB command window "Warning: LP failed to converge". This happens when the MATLAB provided "linprog,m" optimization routine fails to find a solution, even though the problem is feasible. The AFSimPlan software handles this by either discarding the current random sample and getting a new one, or perturbing the inputs and trying again. The problem with the LP seems to occur about once out of every 50,000 LPs. During a 10 sample stochastic evaluation, 200 LP's are run. Therefore, in a long "Improve Plan" run of say, 300 improvements, you have a good chance of seeing one of these warnings.

Level 2 Analysis

This portion of the application is invoked by typing "level2analysis" at the MATLAB command prompt, or by pressing Move to Level 2 from the Level 1 control panel. The control panel shown in Figure A5 below will be displayed.

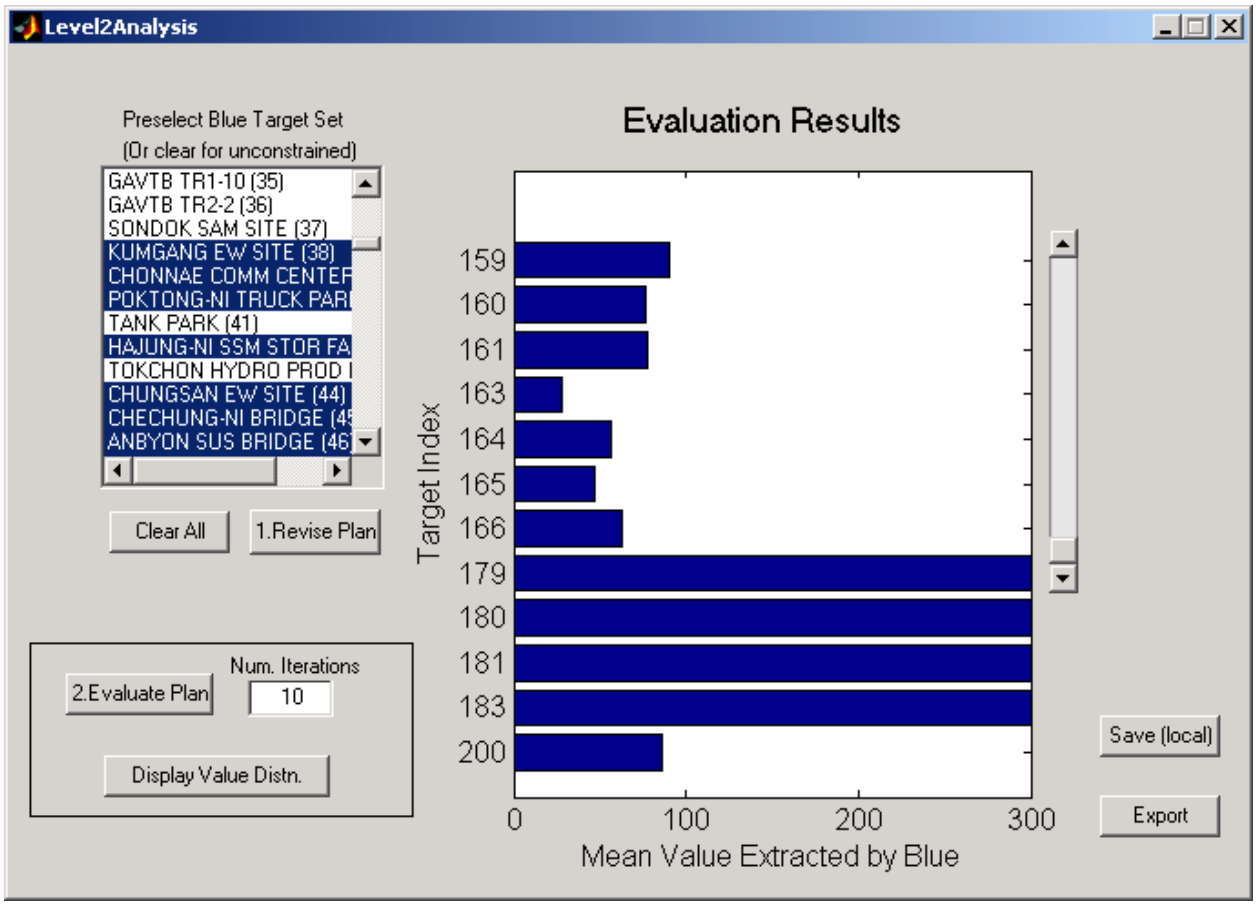


Figure A5. Level 2 Controls

The Level 2 Analysis is constrained by the Level 1 Blue plan determined during the Level 1 Analysis above. The first step is to find a conforming Level 2 Plan.

1. Revise Plan (Button). This will create the Level 2 Plan. If no targets are preselected in the list box above the button, the algorithm will select its own best set of high value targets.

> *Preselect Blue Target Set* (list box). Optionally, select the targets to be hit by clicking on each target (use CTRL-left mouse button)
>
> *Clear All* (Button). Clears all selections in the list box

After the plan has been revised, the Evaluation Results will be cleared until a new evaluation occurs (requires only a few seconds on the benchmark machine).

2. Evaluate Plan (Button). This invokes the Level 2 Stochastic Evaluator, providing sample values for the current period. Upon completion, the results are displayed by target in *Evaluation Results*. The function makes use of *Num. Iterations*.

> *Num. Iterations* – the number of samples taken to make the evaluation
>
> *Display Value Distn*. (Button) - provides a cumulative probability distribution of the current net Blue sample values analogous to Figure A4 above.

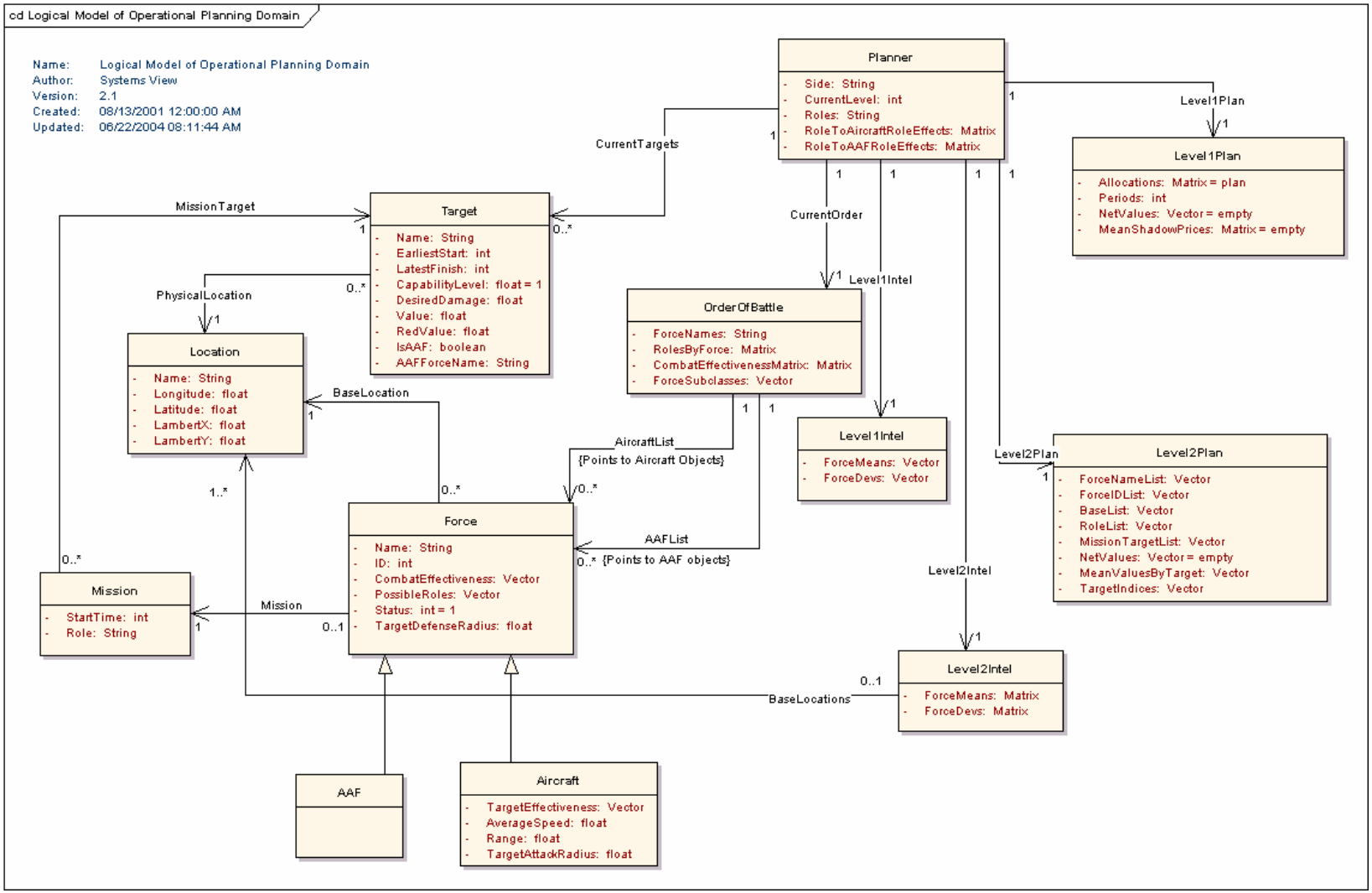
This evaluation is significantly faster than the similar Level 1 evaluation (approx. 30 seconds on the benchmark machine for 10 iterations)

Save (local) (Button) - the same operation as in Level 1 Analysis.

Export (Button) - the same operation as in Level 1 Analysis.

Appendix B: Domain Model

The domain model diagram shows a UML specification (Booch, et al., 1999) for the domain objects implemented in MATLAB. The domain objects contain all of the problem domain related data in a standard format for use by the various algorithms. The implementation makes use of the "object-oriented" features of MATLAB. The domain model shown below is an image of the UML specification in the software design tool, Enterprise Architect (filename af.eap). A free, read-only version of the software, as well as a trial version, is available at the company website: http://www.sparxsystems.com.au/ .


cd Logical Model of Operational Planning Domain
Name: Logical Model of Operational Planning Domain
Author: Systems View
Version: 2.1
Created: 08/13/2001 12:00:00 AM
Updated: 06/22/2004 08:11:44 AM
Planner
- Side: String
- CurrentLevel: int
- Roles: String
- RoleToAircraftRoleEffects: Matrix
- RoleToAAFRoleEffects: Matrix
Level1Plan
Level1Plan
- Allocations: Matrix = plan
- Periods: int
- NetValues: Vector = empty
- MeanShadowPrices: Matrix = empty
CurrentTargets
CurrentOrder
Level1Intel
Level2Plan
Level2Intel
MissionTarget
Target
- Name: String
- EarliestStart: int
- LatestFinish: int
- CapabilityLevel: float = 1
- DesiredDamage: float
- Value: float
- RedValue: float
- IsAAF: boolean
- AAFForceName: String
PhysicalLocation
Location
- Name: String
- Longitude: float
- Latitude: float
- LambertX: float
- LambertY: float
BaseLocation
OrderOfBattle
- ForceNames: String
- RolesByForce: Matrix
- CombatEffectivenessMatrix: Matrix
- ForceSubclasses: Vector
AircraftList
{Points to Aircraft Objects}
Level1Intel
- ForceMeans: Vector
- ForceDevs: Vector
Level2Plan
- ForceNameList: Vector
- ForceIDList: Vector
- BaseList: Vector
- RoleList: Vector
- MissionTargetList: Vector
- NetValues: Vector = empty
- MeanValuesByTarget: Vector
- TargetIndices: Vector
Force
- Name: String
- ID: int
- CombatEffectiveness: Vector
- PossibleRoles: Vector
- Status: int = 1
- TargetDefenseRadius: float
AAFList
{Points to AAF objects}
Mission
- StartTime: int
- Role: String
Mission
Level2Intel
- ForceMeans: Matrix
- ForceDevs: Matrix
BaseLocations
AAF
Aircraft
- TargetEffectiveness: Vector
- AverageSpeed: float
- Range: float
- TargetAttackRadius: float

List of Symbols, Abbreviations, and Acronyms

AAF – Anti-Aircraft Forces (e.g. SAM, Patriot)

AFSimPlan – Air Force Simulation Planning -the name of the software implementation of this research

COA – Course of Action

JFACC (Joint Force Air Component Commander) a DARPA research program

LP – linear program - a mathematical modeling formulation

MATLAB – the scientific programming software language used to implement the algorithms

SAM – surface to air missile

UML – Unified Modeling Language