16th ICCRTS:

Collective C2 in Multinational Civil-Military Operations

"An Optimization-based Multi-level Asset Allocation Model for Collaborative Planning"

Modeling and Simulation

Experimentation, Metrics, and Analysis

Collaboration, Shared Awareness, and Decision Making

Students: Xu Han, Suvasri Mandal, Huy Bui, Diego Fernando Mart ńez Ayala, David Sidoti, Manisha Mishra

E-mail: {xuh06002, sum07002, hnb05001, dfm08004, sidoti, manisha.mishra}@engr.uconn.edu

Prof. David L. Kleinman

E-mail: dlkleinm@nps.edu

Prof. Krishna R. Pattipati*

University of Connecticut

Dept. of Electrical and Computer Engineering

371 Fairfield Way, U-2157

Storrs, CT 06269-2157

FAX: 860-486-5585

Phone: 860-486-2890

E-mail: krishna@engr.uconn.edu

*To whom correspondence should be addressed: krishna@engr.uconn.edu

An Optimization-based Multi-level Asset Allocation Model for Collaborative Planning*

Xu Han[1], Suvasri Mandal[1], Huy Bui[1], Diego Fernando Mart ńez Ayala[1], David Sidoti[1], Manisha Mishra[1], Krishna R. Pattipati[1] and David L. Kleinman[2]

[1]Univ. of Connecticut, Dept. of Electrical and Computer Engineering, Storrs, CT 06269

[2]Naval Postgraduate School, Dept. of Information Sciences, Monterey, CA 93943

ABSTRACT

Motivated by the Navy's emphasis on networked planning capabilities in maritime operations centers (MOC), we have developed an agent-based multi-level resource allocation model that takes high level commands from the human planners and then dynamically allocates the lower-level assets and processes tasks to accomplish the mission objectives. The agent-based model supports a controllable, multi-player, real time collaborative planning environment in a Windows environment. The architecture allows for adding constraints on and manipulations of organizational structures, such as authority, information, communication, resource ownership, task assignment, as well as mission and environmental structures. The planning problem is formulated as a multi-level optimization problem of minimizing the overall difference between the human specified performance measures and expected performance measures which are evaluated based on how well the assigned resources match the required resources, subject to a number of real-world planning constraints on assets. We applied a Dynamic List Planning algorithm (DLP) to solve the intractable multi-level resource allocation problem. The near-optimal DLP method can generate high-quality solutions in seconds compared to days taken by the branch-and-bound-based search methods.

Keywords: Collaborative planning, Dynamic List Planning, Maritime operations centers, Multi-level asset allocation problem

I. INTRODUCTION

Motivation

The Navy's new concept of incorporating maritime headquarters with maritime operations centers (MOC) emphasizes standardized processes and methods, centralized assessment

* This work was supported by the U.S. Office of Naval Research under Grant N00014-09-1-0062

and guidance, networked distributed planning capabilities, and decentralized execution for assessing, planning and executing missions across a range of military operations [1]. The planning process is informed by guidance from higher-level headquarters and the assessment process. It is collaborative both vertically, with higher-level headquarters and lower-level subordinates, and horizontally, with other MOCs and joint components. The maritime planning processes focus on the desired objectives and operational effects specified by higher-level headquarters' guidance. A MOC does not "own" any forces, but rather gives directives to subordinate commands and forces. The primary tool for the MOC will be the collaborative information environment (CIE). Important to effective execution is operational environment awareness, horizontal and vertical integration with other commands and continuous assessment. Coordination and collaboration play a key role in the MOC's distributed planning environment [2].

Organizational decision-making, including the process of recognizing when and how to adapt, is a complex, knowledge intensive process [4]. In order to study the coordination behavior among planners and to mimic operational planning processes in the MOC, we have developed an agent-based multi-level resource allocation model that takes high-level commands from the human planners and then dynamically allocates the lower-level assets and schedules tasks to accomplish the mission objectives. It is an interactive decision aid to facilitate the planning process in MOC architecture. The overarching objective is to have an analytical framework or paradigm that facilitates experimentation, analytical/normative modeling of task-asset allocation, supports collaboration and coordination, and software agents to solve the task-asset allocation problem at the subordinate task force (STF) level.

This paper focuses on the human-synthetic agent system which enables human Decision Makers (DM) to perform creative tasks at the operational level planning. Specifically, humans allocate sub-ordinate task forces to tasks and make decisions to adapt organizational elements, while the synthetic agents provide information and decision support to human DMs by allocating resources at the tactical level based on the commander's intent. The latter involves performing tactical resource allocation subject to mission constraints. We consider three components that have been proven to have relevant impact on operational level planning processes:

1) Mission environment;
2) Organizational structure; and
3) Supporting-supported relationships within the mission-specific organizational structure.

A human-in-the-loop planning tool is instrumental in facilitating collaborative planning.

Related Research

Our foray into distributed planning started with the MOC–oriented human-in-the-loop planning experiment (MOC-1) conducted at NPS in March 2009. This experiment was designed to examine alternative structures and processes for coordination of planning

activities among three key cells: Future Operations (FOPS), Current Operations (COPS), and Intelligence Surveillance and Reconnaissance (ISR) necessary to operationalize a plan generated by a Future Plans Cell. The FOPS cell drove this experiment – its goal was to develop the "plan" for allocating assets across a number of interdependent future tasks. As part of this experiment, we developed two optimization-based modules that focused on the Future Operations (FOPS) cell's planning activities and Current Operations' (COPS) Risk Analysis. The FOPS Planning Module is a decision aid that presents the planners with *N*-best asset packages that would meet individual task requirements, while maximizing task execution accuracy, to assist the human player in generating an effective (measured in terms of task accuracy) and efficient (measured in terms of task completion times) plan. The ISR and COPS cells supported this activity by obtaining and providing relevant task information needed by FOPS to construct the plan. We also proposed an optimization-based scheduling algorithm that was used by experiment designers to set the conditions for the mission planning activity (e.g., asset types and numbers, task requirements and asset capabilities), and to assure that the tasks presented to the human planners would indeed be achievable to a specified degree of accuracy. Current Operations (COPS) Risk Analysis module was also implemented to assist COPS players on the consequences of redirecting assets from an ongoing task to perform ISR tasks and unforeseen tasks requiring immediate attention [1][3].

The problem of allocating assets to tasks, in its most simplified form, is a well-known assignment problem [5]. When there are a large number of assets available and if they can be combined into different asset packages, it is related to the Cutting Stock problem [6]. There exist many optimization tools that provide solutions to such allocation problems (e.g. Lpsolver, ILOG CPLEX, LINGO, etc.). Motivated by the hierarchical organization of assets [4][7][8][9], the present paper solves a multi-level asset allocation problem, which generates the assignment relationships between the each level's assets and the tasks.

Organization of the Paper

This paper is organized as follows. Section II explores the components of Operational level planning, and introduces a mission environment model, the basic concepts in multi-level asset-task modeling paradigm, organizational hierarchy and supporting-supported relationships. The allocation problem for MOC-2 experiment is formulated in Section III. Herein, the optimization objective is one of minimizing the difference between the assigned accuracy based on the assignment array and the desired accuracy specified by human players. A heuristic approach, termed the Dynamic List Planning method, is applied to the problem in Section IV. The experiment design and results are presented in Section V. Finally, the paper concludes with a summary of key findings and future research directions in section VI.

II. Components of Operational Level Planning

Mission Environment

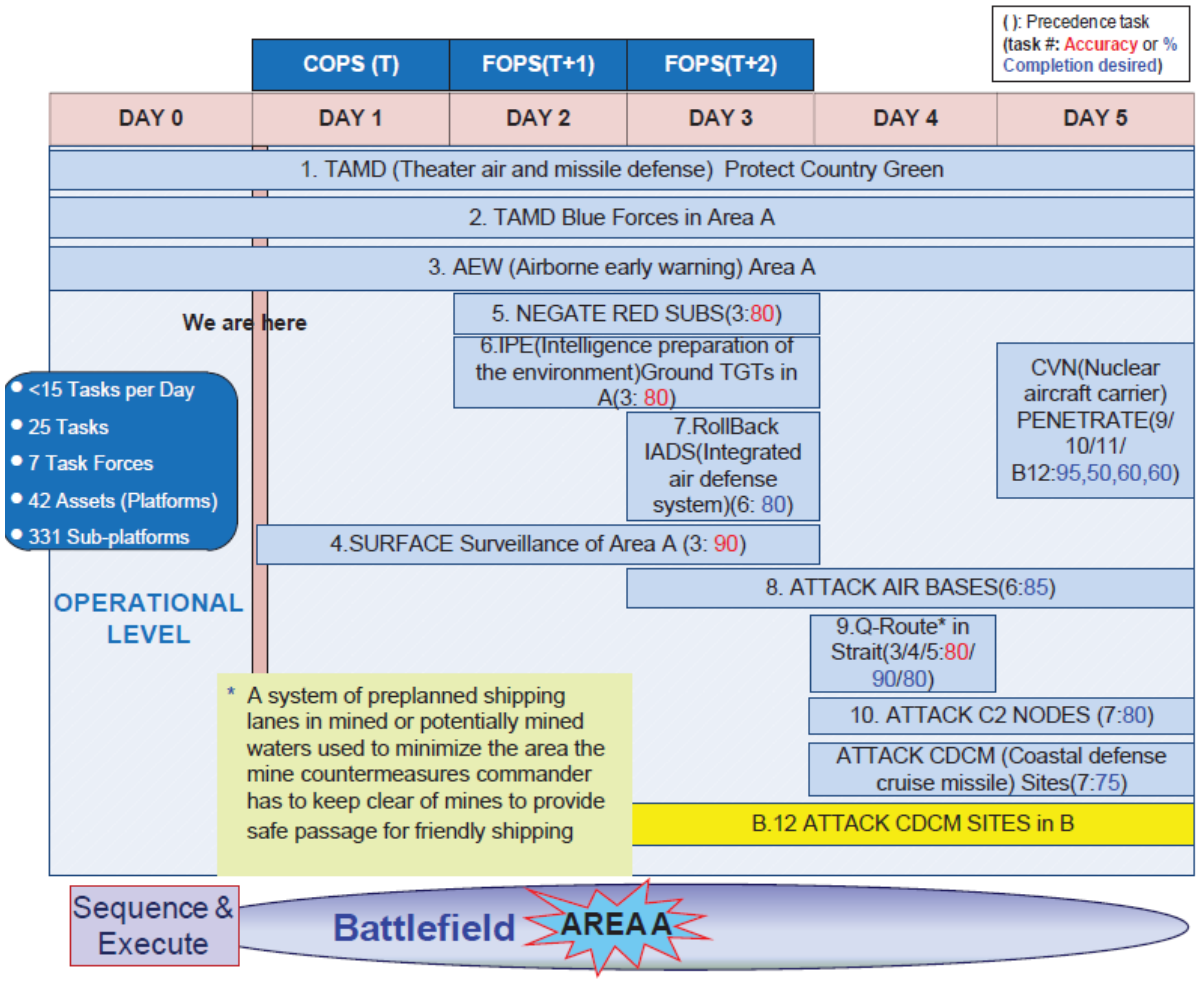


Figure 1 Mission Task Graph for Area A

In the MOC experiment [4] conducted at Naval Postgraduate School (NPS), there were two cells: the FOPS (Future Operations) and the COPS (Current Operations). The experiment was to build an effective plan by allocating Task Forces (TF) to a task, monitoring the performance of the task and then re-planning on the next day for the ongoing task based on the performance at the end of previous day. The experiment was conducted with six teams for a six day planning time window starting with Day 0 and ending with Day 5. There were four FOPS players in each team. The experiment was performed under two conditions: integrated team where the teams planning for Area A were aware of plans for Area B and isolated team where the two teams were planning separately. Both teams were given the agent-based software decision support tool that encouraged coordination (integrated) or reduced coordination (isolated). The FOPS team worked on developing a plan for both geographical Areas A and B for time blocks of Day (T + 1) and Day (T + 2), whereas the COPS cell monitored the plan for current play session (i.e., Day T) and send status updates to the FOPS as situational reports for the assets and the task which represented the true information from the battlefield. The planning for each team was further subdivided into subteams, where each FOPS player

was either responsible for Area A or for Area B tasks. Since they worked with the same set of assets for different planning blocks and as there may not be adequate assets to complete all tasks with desired accuracy, they needed to coordinate to generate a feasible plan to maximize the mission's average task accuracy. Figure 1 shows the task graph for Area A. A similar graph is specified for Area B.

Multi-level Asset Task Modeling Paradigm

Our experimental model consisted of the following entities.

Human-Players (Decision Makers-DM): The human players set the overall desired accuracy or % completion for each task T_i and assign Task Forces (TFs) to accomplish the tasks. Each human player is assigned a list of tasks for which they are responsible. The FOPS players are grouped into a team of four players, where each player is responsible for planning on day (T+1) or day (T+2), Area A or Area B. T varied from 0 to 5.

Tasks: A task, which is derived from mission decomposition, is an activity that requires relevant resources to be processed. The tasks are carried out by a DM under certain mission objectives. In our model, we characterize a task i (i = 1,2,…,I), where I is the total number of tasks, by specifying the following attributes:

1) $t_{s,i}$ = start time of task i ;
2) $t_{p,i}$ = processing time of task i;
3) ρ_i = priority of task i.
4) $\underline{loc}_i$= [x_loc_i, y_loc_i] = x-y coordinates of the geographic location of task i;
5) $\underline{R}_i$ = [R_{i1},. . . , R_{iM}] = resource requirement vector, where R_{im} is the number of units of warfare area m (m = 1, 2, . . , M) required for successful processing of task i.

Task Force (TF): A Task Force (TF) is a physical entity with given asset capabilities and is used to process the tasks. Each Task Force k (k = 1,2,…K), is a composition of assets, with the following attributes:

1) $\underline{loc}_k$ = [x_loc_k, y_loc_k] = x-y coordinates of the geographic location of TF k;

2) $\underline{rng}_k$ = [rng_{k1},…, rng_{kM}] = coverage range of warfare area m.

Assets: An asset A_{kl} (l = 1, …, L) is a physical entity with given resource capabilities and is a decomposition of TF k. For each asset, we define the attributes as follows:

1) $\underline{r}_k$ = [r_{kl1},. . . ,r_{klM}] = resource capability vector, where r_{klm} is the number of units of warfare area m possessed by asset A_{kl};
2) $\underline{max\_warfare\_tasks}_{kl}$ = [$max\_warfare\_tasks_{kl1}$,…,$max\_warfare\_tasks_{klM}$] = concurrent tasking constraints based on the warfare areas, that is, an asset A_{kl} may be involved in at most $max\_warfare\_tasks_{klm}$ tasks in a given warfare area m.

Resource Quantity: A resource quantity p (p= 1,…, P) is a virtual entity with given capabilities α_{klmp} in a particular warfare area m and is a subdecomposition of an asset A_{kl}.

The number of resources allocated to a particular task in a particular warfare category was limited by the tasking constraint.

Warfare Area: Warfare areas represent functional categorization of resources, e.g, Command and Control (C^2), Strike, Intelligence Surveillance and Reconnaissance on ground. Categorization of resources into warfare areas provides a common language to specify task requirements and asset capabilities. That is, each task is specified by resource requirements in each functional warfare area and each asset provides resource capabilities (including zero) in each warfare area.

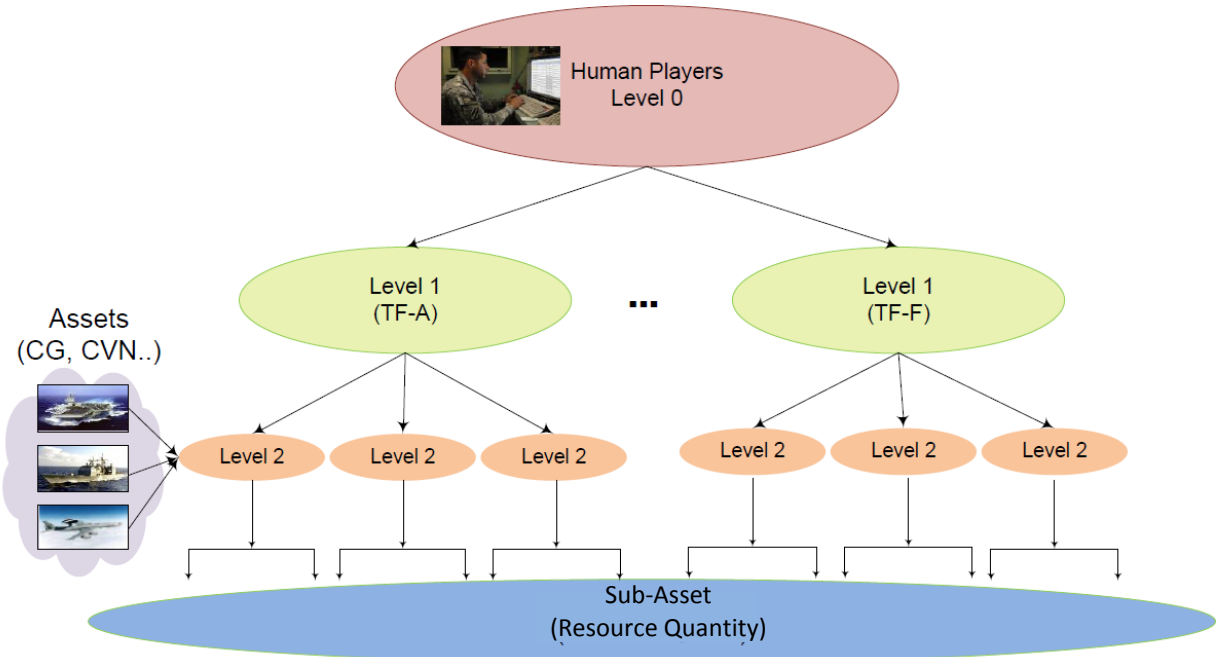


Figure 2 The Organizational Hierarchy for MOC

Organization Hierarchy

In this experiment, we assumed an organizational hierarchy. Level 0 is the top level (Human Player/DM); Level 1 comprises the TF; level 2 are the assets within individual TFs. DM indicates the command cell at root level 0 (e.g., staff organized as a MOC), gives orders and assigns tasks to task forces (technically via his subordinate commanders) at level 1; these indicate the set of assets at level 1 that receive their orders from the MOC DM. These are the Task Forces immediately subordinate to the MOC, e.g. an ESG, CSG, destroyer squadron, etc. A commander at level 1 (including respective staff) gives orders and assigns tasks to forces at level 2. The set of assets at level 2 receive their orders from commanders at level 1. The assets at level 2 are the decomposition of an asset at level 1 into components. Thus, each level 1 asset is an asset grouping, e.g., CSG (TF-A), and each level 2 asset, e.g., CG, DDG is an individual asset, etc. The individual asset at level 2 can be further subdivided into virtual assets or resource quantity. The resource

quantities are allocated to tasks subject to the mission constraints. Thus, an asset at level 2, e.g., CG, can be involved in a STRIKE mission as well as a command and control mission simultaneously as well as a task can have a CG assigned to it and a CVN assigned to it simultaneously. Figure 1 shows the overall mission plan or course of action (COA) that is being operationalized by the MOC for Area A. Figure 2 shows the organization hierarchy used in the experiment.

Supporting-Supported Relationships

In our previous research [10], it was stated that a single asset could only be assigned to a single (responsible) task, although a task could be assigned to more than one asset. In order to have an asset assigned to more than one task, it is necessary to introduce the concept of supporting-supported, or a primary and secondary asset (or TF). There is only one primary TF or asset, but there could be several secondary TFs. At level 0, DM0 could assign a task T_4 to A_3 as the primary (supported) asset, but could also assign the same task T_4 to A_2 as a supporting asset for specific warfare areas. Moreover, DM1 will have at his disposal (or can request) assets used by DM0 when he allocates the task T_1 to asset A_3's resources quantities. This leads to inter-asset coordination and collaboration in processing task T_4. Although the primary Task Force might be the first choice to get allocated for the responsible task, the supporting Task Force could always come to the rescue for the warfare categories which might be out of range or unavailable to the primary Task Force. The supporting relationships could be changed from one decision epoch to the next, depending on other (unanticipated) events. The primary or the supported Task Force for any task should be fixed for that task throughout the experiment [4].

III. Agent Problem Formulation

The mission can be decomposed into a set of tasks that entails the use of relevant resources and is carried out by a Task Force (TF) or a group of TFs. Consequently, the mission goal can be achieved by accomplishing all the tasks in a sequential or parallel manner according to the interrelationships in the task graph. In the MOC-2 experiment [4], the human players specify the desired tasks performance, which is defined as either in terms of task accuracy or % task completion. If the asset allocation is such that all the tasks can reach the desired performance, then the planning objective is said to have been maximized. The task requirement is a vector of resource categories. Ideally, if assigned resources match the required resources perfectly for all the warfare areas, it reaches the desired performance level. If there is a mismatch between task requirements and allocated resources, due to Task Forces geographic coverage constraints or scarcity of assets, task performance is less than desired. We adopt the definition of task accuracy (*Acc*) in [1] as follows:

$$Acc(i) = \sqrt[|\gamma(i)|]{\prod_{m\in\gamma(i)} \min\left\{ Acc_{upper}, \sum_{k\in prmr(i)\cup scnd(i)} \sum_{l\in L(k)} \sum_{p\in P(k,l,m)} \frac{\alpha_{klmp} z_{iklmp}}{R_{im}} \right\}} \qquad (1)$$

where m is the index of the warfare area; k is the index of the Task Force; $\gamma(i)$ denotes the

set of warfare areas which task i requires, i.e.,($R_{im} > 0$); $|\gamma(i)|$ is the cardinality of $\gamma(i)$; Acc_{upper} denotes the upper bound on accuracy; α_{klmp} denotes the number of resources the resource quantity/sub-asset owns; z_{iklmp} is the assignment variable denoting the status of assignment of task i to Task Force k's level 2 asset l in warfare area m with resource quantity index p, e.g., Figure 3 shows an example of multi-level asset-task allocation.

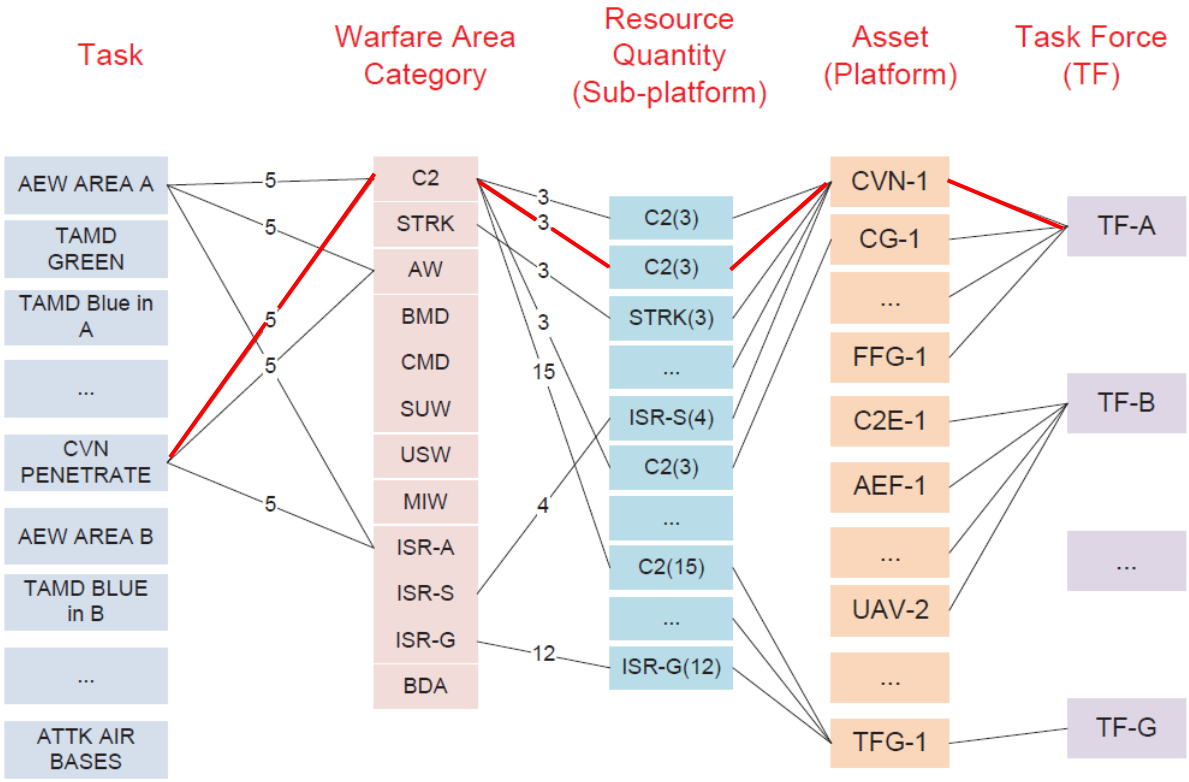


Figure 3 Multi-level Asset-Task Allocation

Here, 3 units of TF-A's t CVN-1's capabilities in C2 warfare area are assigned to the task CVN PENETRATE; $prmr(i)$ is the primary (supported) Task Force for a task T_i ; $scnd(i)$ is the set of secondary (supporting) Task Forces for a task T_i ; l is the index of the level 2 asset e.g. CG; $L(k)$ is the set of level 2 assets belong to TF k, e.g., CG-1 that TF-A; p is the index of the resource quantity; $P(k, l, m)$ is the set of resource quantities that belong to asset A_{kl} in a warfare area m; R_{im} is the task T_i requirement for warfare area m. It is evident that accuracy is a monotone increasing function of assigned resources, i.e., as more resources are assigned to a task, higher is the execution accuracy of the task. If any warfare area's resources are missing, the task has zero accuracy and thus cannot be processed.

The Acc_{upper} limits the maximal impact of one warfare area on the task's accuracy. The upper limit on accuracy in each warfare category is usually set as 100 % or higher e.g., (200% in MOC experiment). Note that a task's performance cannot be improved by adding only one kind of warfare resource. Instead, a balance on all the resource categories (warfare areas) is preferred.

Another criterion of tasks performance is percentage completion. At any day *T*, let *C*(*T*) be the percent (%) completion. Then,

$$C(T) = \begin{cases} 0 \quad \text{if } T = -1 \\ C(T-1) + 100\dfrac{Acc(T)}{t_p} \quad \text{otherwise} \end{cases} \tag{2}$$

where *C*(*T*) is the % completion of the task on day *T*. Note that this is a purely linear charging/completion model with completion rate $100 * Acc(T)/t_p$: so if $Acc(T) = 1$, $t_p = 1$, and $C(T-1) = 0 \Rightarrow C(T) = 100$. Note that once desired percentage completion is given, the corresponding percent accuracy can be calculated easily.

We formulate the asset-task allocation problem as one of minimizing the weighted (by task priorities) average difference between the tasks' desired performance and the actual performance based on the resource allocation. Formally,

$$\text{obj: } \min \sum_{i=1}^{I} \rho_i \frac{1}{|\gamma(i)|} \sum_{m \in \gamma(i)} \left| \sum_{k \in prmr(i) \cup scnd(i)} \sum_{l \in L(k)} \sum_{p \in P(k,l,m)} \frac{\alpha_{klmp} z_{iklmp}}{R_{im}} - Acc_d(i) \right| \tag{3}$$

Note that there exist constraints at the resource, asset and TF levels. Define the asset-to-task per warfare area assignment as

$$y_{iklm} = \begin{cases} 1 \quad \text{if } \sum_{p \in P(k,l,m)} z_{iklmp} > 0 \\ 0 \quad \text{if } \sum_{p \in P(k,l,m)} z_{iklmp} = 0. \end{cases} \tag{4}$$

Note that the function y_{iklm} (z_{iklmp}) is a nonlinear function of z_{iklmp}. Since the variables are binary, we can write the relationships between y_{iklm} and z_{iklmp} as

$$\frac{\sum_{p \in P(k,l,m)} \alpha_{klmp} z_{iklmp}}{r_{klm}} \leq y_{iklm} \tag{5}$$

The multi-level asset planning problem can be thus formulated as:

$$\text{obj: } \min \sum_{i=1}^{I} \rho_i \frac{1}{|\gamma(i)|} \sum_{m \in \gamma(i)} \left| Acc_d(i) - \sum_{k \in prmr(i) \cup scnd(i)} \sum_{l \in L(k)} \sum_{p \in P(k,l,m)} \frac{\alpha_{klmp} z_{iklmp}}{R_{im}} \right|$$

$$
\begin{aligned}
s.t\ & \frac{\sum_{p\in P(k,l,m)} \alpha_{klmp} z_{iklmp}}{r_{klm}} \le y_{iklm} \quad \forall\ i, k, l \text{ and } m; \\
& \sum_{m=1}^{M} y_{iklm} \le M x_{ikl} \quad \forall\ i,k,l; \\
& \sum_{k\in prmr(i)\cup scnd(i)} \sum_{l\in L(k)} x_{ikl} \le \text{max\_assets} \quad \forall\ i\ ; \\
& \sum_{i=1}^{I} x_{ikl} \le \text{max\_tasks} \quad \forall\ k \text{ and } l; \\
& \sum_{i=1}^{I} y_{iklm} \le \text{max\_warfare\_tasks}_{klm} \quad \forall\ k,\ l \text{ and } m; \\
& z_{iklmp} = 0 \quad \forall\ i,\ \ k \in scnd(i),\ \ l \in L(k),\ m \in \phi(k,i),\ p \in P(k,l,m)\ ; \\
& z_{iklmp} = 0 \quad \forall\ k,\ 1 \le l \le L(k),\ \forall\ m,\ p \in P(k,l,m),\ i \in \zeta(k,m)\ ; \\
& x_{ikl} \in (0,1), z_{iklmp} \in (0,1), y_{iklm} \in (0,1).
\end{aligned}
\tag{6}
$$

where $\phi(k, i)$ is the set of resource categories that secondary TFk cannot support; $\zeta(k, m)$ is the set of tasks T_i which is not in the range of the asset A_{kl} for a resource category m; $Acc_d(i)$ is the desired accuracy for the task.

IV. Dynamic List Planning Method

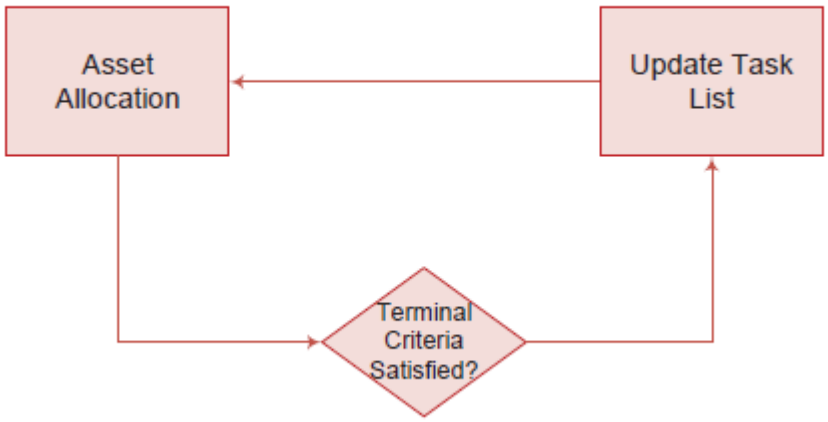


Figure 4 Illustration of Dynamic List Planning Method

The planning problem is a multi-level asset-task assignment problem. The number of inequality constraints is $I * K * L(k) * |\gamma(i)| + I * K * L(k) + I + K * L(k) + K * L(k) * |\gamma(i)|$, is approximately 3000 and the number of decision variables are approximately 2000 for MOC-2 experimental scenario, which makes it impossible to find the optimal solution in a few seconds. Instead, efficient sub-optimal solutions are preferred.

Here, we proposed an approach that provides very good quality near-optimal solutions,

termed the Dynamic List Planning method. The key idea is to plan tasks sequentially based on dynamic task priorities, which are adjusted based on the current resource distribution. The main flow of the Dynamic List Planning method is shown in Figure 4. At the beginning, there is a pool of tasks which needs to be processed. The algorithm deals with one task a time according to the order in the task list. After assigning resources to the top task in the list, the algorithm reprioritizes the tasks in the list based on their requirements and the current assignment.

The dynamic rank (priority) of a task is defined as

$$rank(i) = \begin{cases} \rho_i \left| Acc(i) - Acc_d(i) \right| & \text{if } Acc(i) < Acc_d(i) \\ 0 \quad \text{otherwise} \end{cases} \tag{7}$$

Note that $Acc(i)$ is increased when more resources are assigned. Therefore, for two tasks with the same priorities, the one with inadequately assigned resources will be listed at the top. On the other hand, two tasks with the same difference in accuracies, the one with a higher priority will be selected for the next planning phase. Our asset selection involves two criteria. The first one evaluates the accuracy assuming that all the allowable resources from the asset are allocated, and chooses the asset that provides the maximum accuracy. The second one evaluates the desirability of Asset A_{kl} using:

$$\sum_{m \in \gamma(i)} \left(Acc_d(i) - \frac{\min(Acc_d(i) R_{im}, \overbrace{\sum_{p \in P(k,l,m)} \alpha_{klmp}}^{\text{resources from Asset } A_{kl}} + \overbrace{\sum_{k \in prmr(i) \cup scnd(i)} \sum_{\substack{u \in L(k) \\ u \neq l}} \sum_{p \in P(k,u,m)} \alpha_{kump} z_{ikump}}^{\text{already assigned resources}})}{R_{im}} \right) \tag{8}$$

In the asset selection, the first criterion is employed. If more than one asset is needed, the second criterion is applied. After selecting the most suitable asset A_{kl}, the resource allocation problem is solved via:

$$\begin{aligned} &\text{obj: } \min \left| \sum_{p \in P(k,l,m)} \frac{\alpha_{klmp} z_{iklmp}}{R_{im}} - Acc_r(i) \right| \\ &s.t \quad z_{iklmp} = 0 \quad \text{if } k \in scnd(i), \ m \in \phi(k,i), \ p \in P(k,l,m) \\ &\qquad z_{iklmp} = 0 \quad p \in P(k,l,m), \ i \in \zeta(k,m); \ z_{iklmp} \in (0,1), \end{aligned} \tag{9}$$

where $Acc_r(i)$ is the adjusted accuracy, defined as the desired accuracy minus the assigned resources from pervious iteration.

$$Acc_r(i) = Acc_d(i) - \sum_{k \in prmr(i) \cup scnd(i)} \sum_{u \in L(k)} \sum_{p \in P(k,u,m)} \frac{\alpha_{kump} z_{ikump}}{R_{im}} \tag{10}$$

This resource allocation problem can be solved via greedy search. The procedure is shown in Table 1.

Greedy Search:

1. Calculate the remaining resource requirement $R'_{im}=Acc_r(i)*R_{im}$.
2. Scan all the allowable resource quantities, rank them based on $|R'_{im}-\alpha_{klmp}|$ in ascending order
3. Allocate the first resource quantity p' to the task.
4. Calculate $R'_{im} = R'_{im} - \alpha_{klmp'}$. If $R'_{im}\leq 0$, quit; otherwise, go to step 2.

Table 1 Procedure of Greedy search for Resource Quantity Allocation Problem

The procedure for the Dynamic List Planning method is shown in Table 2.

V. Software Design and Algorithm Performances

The agent-based distributed planning model provides a flexible and controllable multi-player, real time planning environment to support laboratory-based empirical experiments in a Windows environment to study the interactions and the planning processes in a MOC architecture. The architecture allows for manipulation of organizational structures, such as authority, information, communication, resource ownership, task assignment, etc., as well as mission and environmental structures. The experimental framework involved a team of

Dynamic List Planning Method:

1. Initially, create the task list with each task T_i's $rank(i)= \rho_i*Acc_d(i)$.
2. Choose the task $T_{i'}$ with the highest rank, call the planning phase
    a. Check each asset's allowable resources.
    b. Rank the assets based on the accuracy if the particular asset assigned to task $T_{i'}$
    c. Select the top asset, if the corresponding accuracy is zero or same as the second asset, go to step 2-d, otherwise, go to step 2-e.
    d. Rank the assets based on Eq. 8, choose the top asset.
    e. Apply the greedy search to find the resource quantity allocation
3. Update the rank of task $T_{i'}$ $rank(i')= \rho_{i'}*|Acc_d(i')- Acc(i')|$ if $Acc_d(i') > Acc(i')$, otherwise, $rank(i')=0$
4. Check the terminal criteria
    a. If all the tasks have reached the maximal number of assets, if true, go to step 5, otherwise, go to step 4-b.
    b. If all the tasks have explored all the allowable assets, if true, go to step 5, otherwise, go to step 2.
5. Add/Drop heuristic process to improve the objective function

Table 2 Procedure of Dynamic List Planning Method

decision-makers (DMs) in a hierarchical or networked organization to simulate a military operation. The players had operational capabilities to plan an operational level task by assigning high level Task Forces that control the necessary resources to execute a set of assigned mission tasks, provided that such task execution does not violate the Task Force's concomitant resource capability thresholds. During the course of the experiment, a DM was able to plan and re-plan ongoing tasks for the next time epoch by modifying the current assignment to increase team performance based on his expertise and the current updates from the battlefield. The designed architecture supported collaboration because the information needed to pursue the joint goal was beyond the capability, knowledge or capacity of any individual player. The architecture also has the facilities to acknowledge the dynamic updates from the battlefield.

Planning Module

The networked distributed coordinated framework involved human players to establish joint or individual commitments to tasks, to monitor the execution of tasks, acknowledge the true information from the battlefield, to broadcast task performance and to re-plan the task, if necessary. The initial static data for the experiment populated the administrator terminal. This experiment had 4 Future Operations (FOPS) terminals, one Current Operations (COPS) terminal and an experimenter or administrator terminal. Figure 5 shows the block diagram of the planning module used in the 2010 MOC-2 experiment.

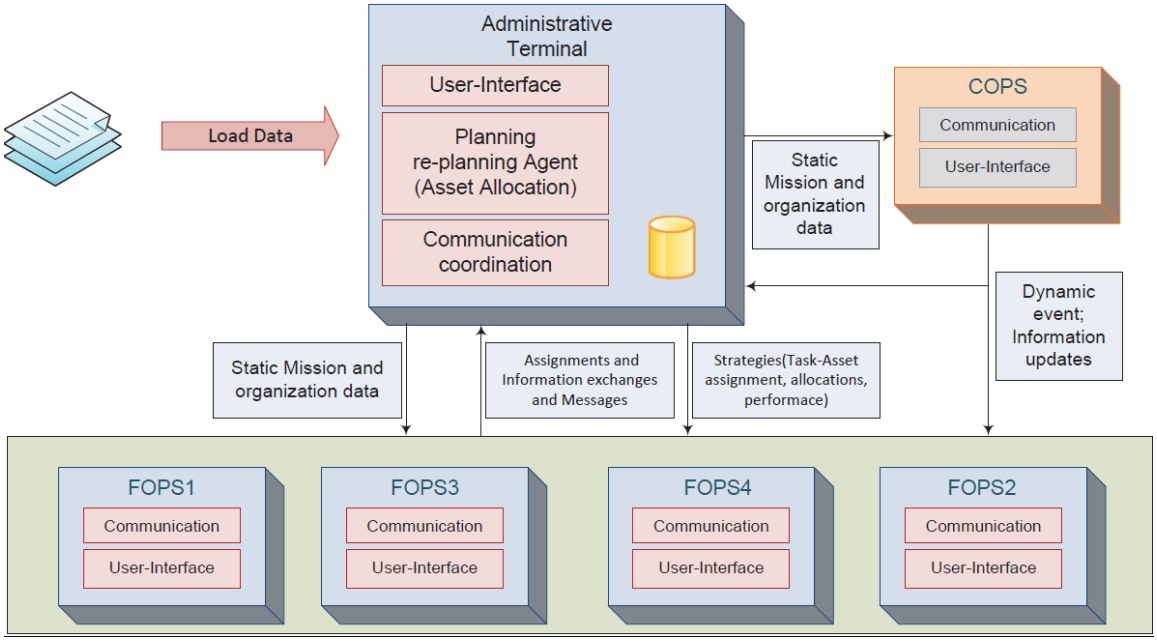


Figure 5 Distributed Planning Module Architecture

Multi-Level Asset Allocation Model: Experimental Setup

The multi-level asset allocation decision support model was used in the MOC-2 experiment conducted at Naval Postgraduate School (NPS). The experiment was set-up

with four Future Operation players (FOPS) and one Current Operation player (COPS) and an administrator to mimic the functionalities of the MOC director. The administrator at the server terminal controlled the experiment, which involved planning in two geographical areas Area A and Area B and four play sessions. The software supported Integrated (centralized) and Isolated (decentralized) team structures. It also supported situational reports (SITREPS) from the battlefield. The software has the following tabs:

(1) Summary Screen: The summary screen provides aggregated planning information to the players on all active tasks for the planning period. It displays the task name, task priority, responsible DM (a responsible Decision Maker is the one who can plan for the tasks for that day; responsibilities may or may not change over time); primary Task Force (TF); secondary TF; criteria (can be % completion or accuracy): desired, expected and actual performance. The active tasks are white, while the inactive tasks are grayed out. Figure 6 shows the summary screen for the planning software.

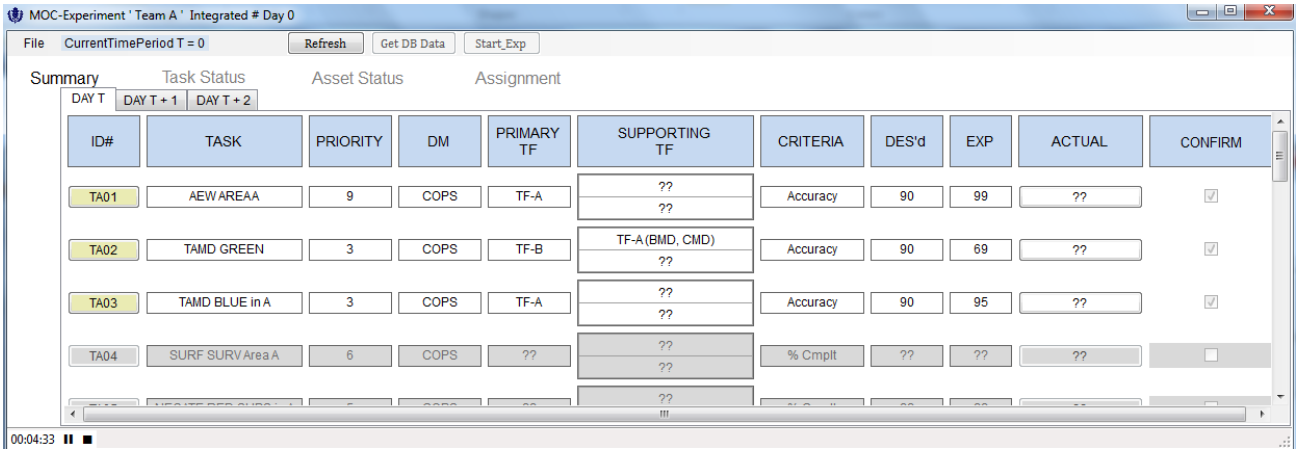


Figure 6 Summary Screen of the 2010 MOC-2 Experiment Software

(2) Task Status Screen: The task status screen shows status (whether the task is started or not) and the requirement vector for all the tasks. It shows the requirement for 100% accuracy criterion. It can be used as a reference by the human planners.

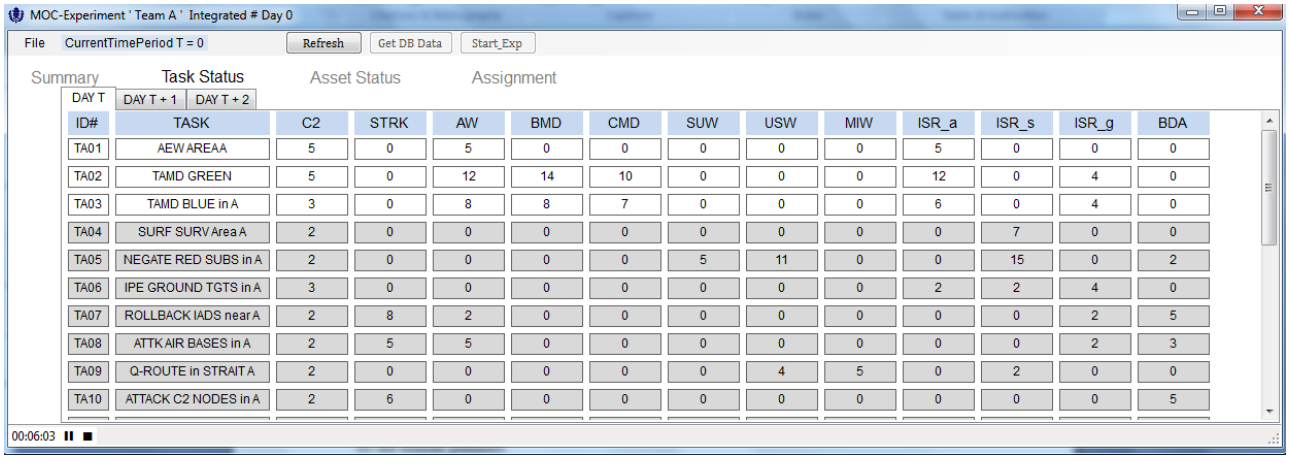
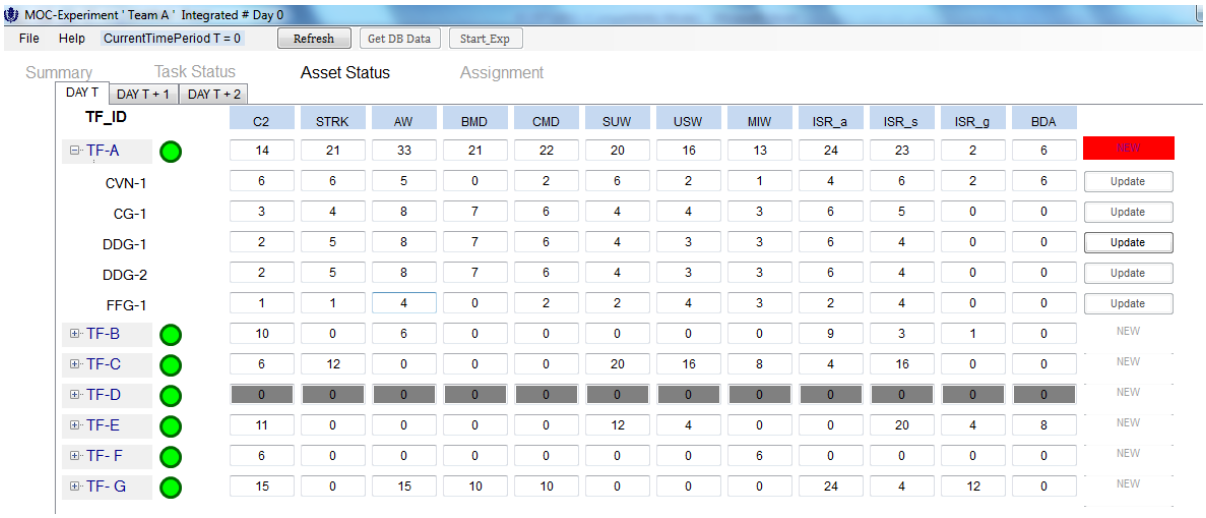


Figure 7 Task Status Screen of the 2010 MOC-2 Experiment Software

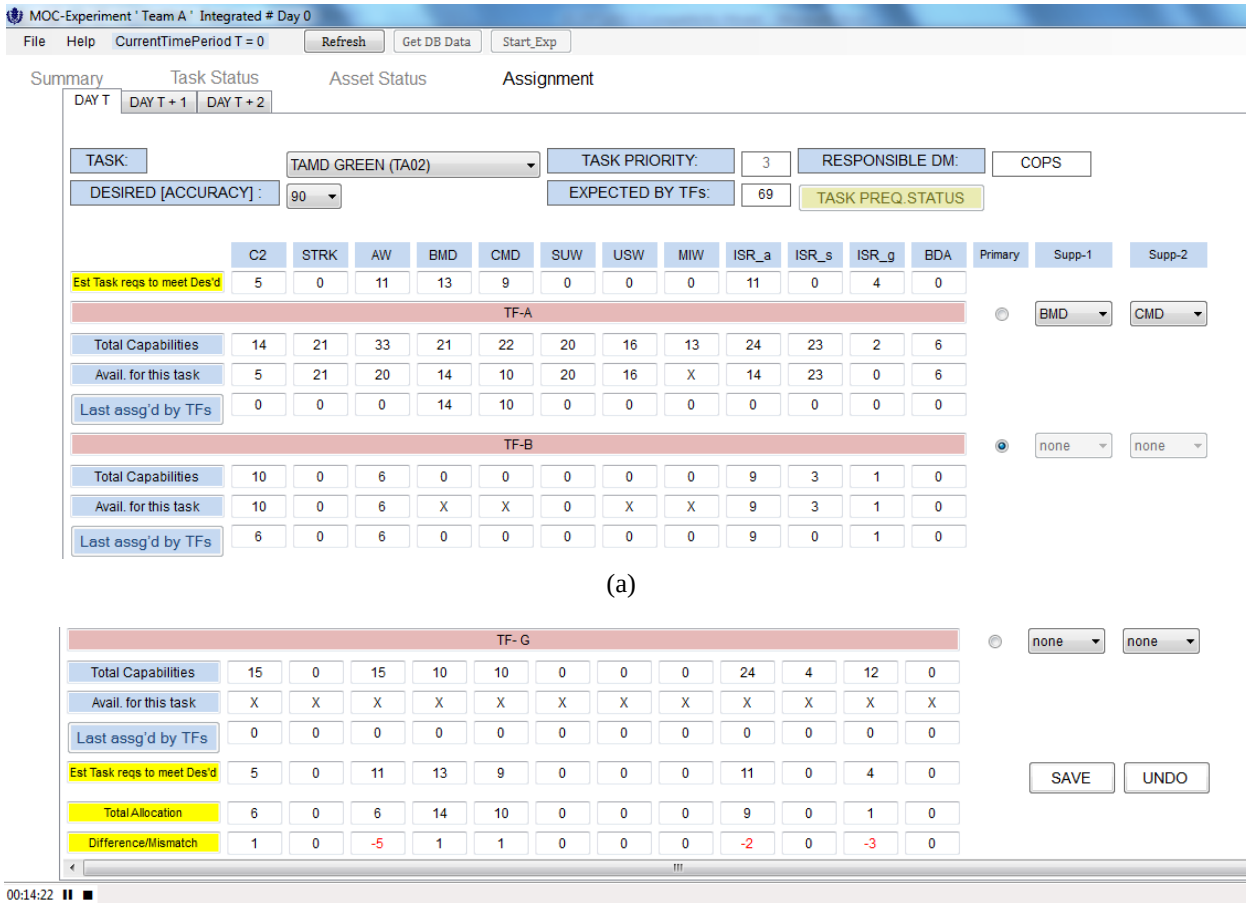
(3) Asset Status Screen: The asset status screen shows the total capabilities of the Task Force. It also displays the individual capabilities of each asset belonging to the Task Force. This display supports the situational reports sent by the COPS. The grayed out assets have either not arrived in the theater or are unavailable. To show the health of the Task Force, we have a stop light besides each Task Force. The messages from the COPS player can be updated using the update button and entering the code sent in by the COPS player.



TF_ID	C2	STRK	AW	BMD	CMD	SUW	USW	MIW	ISR_a	ISR_s	ISR_g	BDA	
TF-A	14	21	33	21	22	20	16	13	24	23	2	6	NEW
CVN-1	6	6	5	0	2	6	2	1	4	6	2	6	Update
CG-1	3	4	8	7	6	4	4	3	6	5	0	0	Update
DDG-1	2	5	8	7	6	4	3	3	6	4	0	0	Update
DDG-2	2	5	8	7	6	4	3	3	6	4	0	0	Update
FFG-1	1	1	4	0	2	2	4	3	2	4	0	0	Update
TF-B	10	0	6	0	0	0	0	0	9	3	1	0	NEW
TF-C	6	12	0	0	0	20	16	8	4	16	0	0	NEW
TF-D	0	0	0	0	0	0	0	0	0	0	0	0	NEW
TF-E	11	0	0	0	0	12	4	0	0	20	4	8	NEW
TF- F	6	0	0	0	0	0	0	6	0	0	0	0	NEW
TF- G	15	0	15	10	10	0	0	0	24	4	12	0	NEW

Figure 8 Asset Status Screen of the 2010 MOC-2 Experiment Software

(4) Assignment Screen: The assignment screen is used by the planners to specify the desired performance based on their area of responsibility; the primary Task Force; the supporting (secondary) Task Force; and the supporting warfare areas that the secondary Task Force will be involved in for all the active tasks. Figure 9 shows the assignment screen for the planning software. The assignment page shows expected performance returned by the planning agent if the plan is submitted. It shows the total capability of each Task Force; the Task Force's available capability (this is the capability which is available for the task selected); last assigned by TF (indicates the allocated capabilities by the TFs to this task). On clicking the last assigned by TF, we can also see the assets, their capability in each warfare category being assigned to the particular task. If in a particular warfare category, the capability is shown as "X", it means that the particular warfare category is out of range for the selected task. The estimated task requirement for the 'desired' is also shown; it corresponds to the requirement vector for the selected task to meet the desired performance. It changes based on the desired performance chosen. The difference and the mismatch at the bottom show the difference in task requirements and the total allocated capabilities from all assigned task forces.



(a)

(b)

Figure 9 Assignment Screen of the 2010 MOC-2 Experiment Software

Results

The Dynamic List Planning method was applied to the experimental mission scenario designed by NPS. Since the constraints exist on task, warfare area, resource quantity, asset and Task Force, the information related to the five levels are required. Thus, the decision variable had high dimensions and the size was up to around 2000 per day. The constraints included the tasks per asset, tasks per asset per warfare area, assets per task, supporting warfare areas and geographic range constraints. Meanwhile, there are up to around 3000 inequality constraints per day. Due to the large number of decision variables and constraints, the allocation problem was formulated into a large scale linear integer programming problem, which cannot be solved in polynomial time (combinatorial optimization problem). The exact algorithm, Branch and Bound algorithm has run time that spans days for a one day plan.

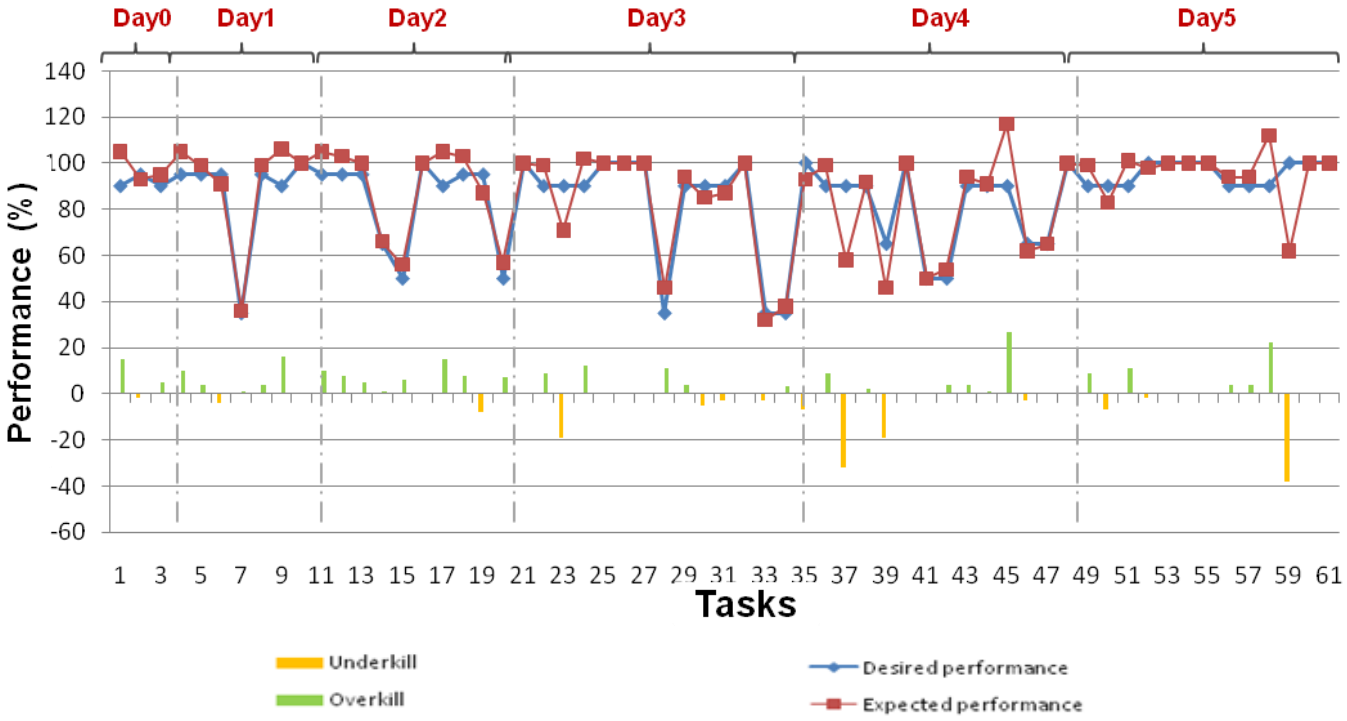


Figure 10 Desired vs. Expected Performance (Accuracy or % Completion)

We conducted computational efficiency and allocation performance tests based on the MOC-2 experiment scenario, set up the desired performance as default values (97% as accuracy or $100\%/t_p$ as % completion) and assigned TFs subject to the TF constraints, which includes: 1) The primary TF for any task should be fixed for that task throughout the experiment; 2) A task can have only one primary TF assigned to it and at most two supporting TF in at most two warfare areas; 3) one TF cannot be primary TF for more than 4 tasks; 4) one TF cannot be secondary TF for more than 3 tasks. We applied our Dynamic List Planning method to 6 planning problems corresponding to the planning sessions Day 0 through 5. The tasks per day varied from 3 to 14. In Figure 10, we represent the desired performances and expected performances. We also compute the difference between the two in terms of *overkill* (excess force to destroy the enemy target) and *underkill* (insufficient force to defeat an enemy). The average deviation from the desired performance is 6.61%, and 2.49% as average *underkill*, 4.11% as average *overkill*. Note that some of the *underkill* result from insufficient resources or asset breakdowns. Besides its near optimality, our approach took less than 10 sec per planning session compared to an estimated 40 days of computing time by exhaustive search.

VI. Summary

This paper provided an overview of a multi-level asset allocation model including: 1) a mathematical formulation of the multi-level asset allocation problem and a near-optimal

algorithm to solve the problem; 2) a distributed networked architecture to support the operational level planning process for different team structures (integrated and isolated). Our mixed initiative asset allocation algorithm operates in real-time subject to a number of realistic planning constraints. The collaborative, multi-player planning software supported the MOC experiments at NPS. We presented this model's application to a mission scenario in the MOC-2 experiment and the performance of the Dynamic List Planning method on the asset allocation problem arising in the MOC-2 planning experiment. The experimental results demonstrate that the Dynamic List Planning method is a highly efficient near-optimal solution for complex multi-level asset allocation problems. The Dynamic List Planning algorithm, when embedded in the collaborative planning software, provided a mixed initiative decision support tool for operational level planning. Our future research will focus on decision support software involving assessment, evaluation and analysis of weather impacts on operational level planning.

REFERENCES

[1] H. Bui, X. Han, S. Mandal, K. R. Pattipati, and D. L. Kleinman, "Optimization-based decision support algorithms for a team-in-the loop planning experiment", *In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, TX, October 2009.

[2] United State Fleet Forces Command. 2007. *Maritime Headquarters with Maritime Operations Center Concept of Operations – An Enabling Concept for Maritime Command and Control.*

[3] S. G. Hutchins, W. G. Kemple, D. L. Kleinman, S. Miller, K. Pfeiffer, S. Weil, Z. Horn, M. Puglisi, and E. Entin, "Maritime headquarters with maritime operations center: A research agenda for experimentation", *In Proceedings of the 14th International Command and Control Research and Technology Symposium*, Washington, D.C., June 2009.

[4] S. G. Hutchins, W. G. Kemple, D. L. Kleinman, S. Miller, K. Pfeiffer, Z. Horn, S. Weil, and E. Entin, "Maritime operations centers with integrated and isolated planning teams", *International Command and Control Research and Technology Symposium*, Santa Monica, CA, June 2010.

[5] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics*, vol. 52, pp. 7-21, October 2004.

[6] D.P. Bertsekas, "Introduction to linear optimization", *Athena Scientific*, Belmont, MA 1997.

[7] Levchuk, G. M., Y. N. Levchuk, J. Luo, K. R. Pattipati, and D. L. Kleinman. 2002. Normative Design of Organizations-Part I: Mission Planning. In *IEEE Trans. Syst., Man, Cybern. A, vol. 32*: 346-359.

[8] Levchuk, G. M., Y. N. Levchuk, J. Luo, K. R. Pattipati, and D. L. Kleinman. 2002. Normative Design of Organizations-Part II: Organizational Structure. In *IEEE Trans. Syst., Man, Cybern. A, vol. 32*: 360-375.

[9] Levchuk, G. M., Y. N. Levchuk, C. Meirina, K. R. Pattipati, and D. L. Kleinman. 2004. Normative Design of Organizations-Part III: Modeling Congruent, Robust, and Adaptive Organizations. In *IEEE Trans. Syst., Man, Cybern. A, vol. 34*: 337-350.

[10] S. Mandal, X. Han, K. R. Pattipati, and D. L. Kleinman, "Agent-based distributed framework for collaborative planning", *IEEE Aerospace Conference*, Big Sky, MN. March 2010.