Automated Alternate Plans Invocation – Oct 2005

Introduction

This document is intended to be a quick-and-dirty guide to the JFORCES alternate plans invocation module. This module uses specific triggers to:

1) Move units from one location to another base on perceived enemy actions
2) Place targets and regions in the ATO generator (start bombing buildups)
3) Advance or retreat friendly units based on health
4) Initiate maritime patrols in response to perceived enemy actions

This document is being written before a presimulation GUI is developed. This is because past users have found it easiest to populate the triggers and responses using SQL queries against scenario generation data. When (if) the GUI is developed this document will be updated.

This document is broken into the following sections:

Algorithm Overview

JFORCES incorporates a alternative action algorithm based on perceived enemy activities and states as well as own-side states. The algorithms used are deterministic, but the data driven approach is general enough that this module has been found useful in a wide range of scenarios. The actions currently supported by this module include:

1) Move units from one location to another base on perceived enemy actions
2) Place targets and regions in the ATO generator (start bombing buildups)
3) Advance or retreat friendly units based on health
4) Initiate maritime patrols in response to perceived enemy actions

This module is a stimulus-response model of command and control operations. This module is limited to the high-level control and does not address detailed command and control issues, such as controlling F-16s in a air intercept. This level of control is handled within the JFORCES framework by tailored modules, such as the Air Defense C2 module for the aforementioned air intercept. The stimuli are simulated commander perception, be it of enemy deployment, activity, or of friendly heath, that causes the simulated commander to order certain responses. There are two major components to this module. The first component is the trigger set.

Triggers

Today, there are few triggers in this module, although JFORCES uses a myriad of triggers in the other control modules mentioned above (e.g. air defense command and control). The two major triggers are:

1) A number of units (enemy or friendly) in a specified region, and
2) my own unit's health.

The number of units in a specified region is the more common trigger. It is useful in responding to enemy actions in an area or reinforcing friendly units. Today, the input is via the a;ternate_plans_criteria database table. Today, there's no GUI, so I'll describe the fields for reference since the algorithm's probably obvious once these fields are understood. The fields are:

- scenario – the scenario to which this rule applies
- side – the side (1 for Red, 2 for Blue) that should use this rule. Note that this should be consistent with the alternate_movement and alternate_ATO tables discussed below.
- check_side – The side to check. This is opposite of the side value if you want to respond to detections of the enemy, the same as the side field if you want to respond to the presence (or lack thereof) of friendly units in the region.
- Target_plat, target_cat and target_medium – These fields should be discussed together because they are all used in the same check. Basically, the system determines whether or not a detection (or friendly unit) within a region should be responded to by using these fields as filters.
    - The target_plat field is the name of a type of asset, e.g. T-72. This must be the exact name as used for the platform in database prototyping. If this is set to an empty string (''), then this check will not be used.
    - The target_cat field is used to define the type of target looked for more generally. Today the only legal values for this field are 'ARMOR' and ''. This test will be skipped if the value is ''. If the value is 'ARMOR' it will include any ground units with a armor hardness of 2.8.
    - The target_medium field is the index of the target category. Legal values are all of the values found in the "Asset Categories" section of ~/common/constants.h. One value of current interest is responding to ships in a region. In this case the value should be 14. A 0 skips this test.
    - The final decision of whether to consider a detection (or unit) in a test is based on the junction of all active tests (again, each test can be skipped by using either a 0 or empty string, as appropriate to the data type).
    - Currently only one value's legal for each criteria. This might be restrictive in some cases (e.g. I don't care if it's a T-72 or a T-80, put enough of them in a spot and I should respond). To date that hasn't been too much of a problem because the target_cat field can handle this case. But if it becomes a problem let me know and it'll be easy to replace the single-value fields with a amplification table.
- The north,east, south and west values define the region within which to test. Currently only

individual rectangular regions can be defined. If more complex regions are required this algorithm can be upgraded to use multiple rectangular regions of arbitrary size.

- The criteria_count and less_or_greater field define threshold values for implementation and test direction. The only legal less_or_greater field are 'LESS THAN' and 'GREATER THAN'. The criteria_count can be any positive integer. Thus together these tests can be set up to cause a response when the number of detected enemy tanks is greater than 500 in a region or when the number of friendly tanks is less than 100.
- Finally, there's the alternative plan. This is the ID of the alternative to be invoked. As said above, these actions can start reinforcements, change ATO targets, or start maritime patrols among other actions. See the next section for details.

Each trigger is single-use; that is each one can only be invoked once in a scenario. No relations are defined between the triggers; they are independent. Thus two triggers can be created with the same criteria to initiate two actions based on the same response. But one action's invocation can not alter the possible invocation of another action.

FYI, the check_response code in sim/initial/alternate_plans.c file is the master routine for the above triggers.

There's an additional trigger type to invoke alternate plans based upon own-unit health. This simple trigger runs at preset scenario times and checks the current status of specified units and responds activates commands in accordance with this health. The controlling routine for this trigger is check_attack_continuation in ~/sim/initial/alternate_plans.c. The two database tables that define the trigger are alternate_movement_2 and alternate_movement_2_list. Alternate_movement_2 defines the plan at a top level using the following fields:

- scenario – The scenario in which to use this rule.
- Time_ - The scenario time (in hours) at which to make the test. This is a one-time check, not a periodic check like the other triggers. This could easily be modified if desired.
- Plan – The plan number to execute. This corresponds to an identically named field in alternate_movement_2_list and alternate_movement_routes.
- Worst_health and Best_health. These values (ranging from 0 to 1+) are ratios between the available assets at the time of the check versus at the beginning of the scenario. At first it might seem odd to have both the best and worst healths, but this approach permits the analyst to define a complex list of alternatives, perhaps performing one activity with a fresh group (which he could define as having a health of .9 to 1.), a second action with a moderately healthy group (with a health he could define as .75 to .9) and a third action with a less healthy group (health <.75). As will be seen in the response section, the action need not be taken by the groups involved in the check; instead another group could be given an order based on the first group's health. This is a means of providing reinforcements.

The second database table, alternate_movement_2_list, lists the group(s) whose health should be checked within the trigger. By storing this information in a second table the analyst has the option of specifying a group of units to check for a trigger instead of being limited to only one. The fields of this table are:

- Scenario – The scenario in which to use this rule.
- Plan – The plan ID. This must match the plan ID in the alternate_movement_2 table.
- Groupname, groupnumber – These are the group identifiers.

Responses

Currently responses include:

1) Move units from one location to another base on perceived enemy actions
2) Place targets and regions in the ATO generator (start bombing buildups)
3) Advance or retreat friendly units based on health
4) Initiate maritime patrols in response to perceived enemy actions

Again, despite the fact that all responses are data-driven, currently there is no GUI to define these operations. So I'll describe the appropriate database tables.

The alternate_movement database table controls the movement of units from one location to another based on detections of the enemy or own-unit state in an area. In addition this table is used for defining maritime patrols, so the description of some fields will be deferred until discussing that option. This table is the only alternate-plans related table that needs to be populated to activate either 1) movement options based on detections of the enemy, 2) reinforcement to shore up weak defensive positions, or 3) generating maritime patrols based on perceived enemy activities. Note that this does not imply that the analyst does not have to populate the particulars including the route definition or maritime patrol staffing as appropriate. The table fields are:

- scenario – the scenario in which to consider this rule.
- Name, id, type – the unit or asset affected by this rule. Note that multiple rows of this table can refer to the same criteria, so a horde of different assets and groups can be rallied to perform varied operations all based on the same trigger. The type can either be 1 if the element to move (or be assigned is an asset, 2 if it's a group. If the type is 1 (asset) then the name and id must match the name and number column in the objectdefn table. If a group then the name and id must match the groupname and groupnumber field in the grouplist table. You'll find items referred to by these two fields throughout the presim GUI.
- Routename, routeid – These define the route to invoke, and agree with the values shown in the route GUI in scenario generation.
- check_id – I now think of this as the alternate plan ID. This value must agree with the alternate_plan column of the alternate_plans_criteria table.
- Posture – This is the posture of the unit when it gets it's reform order. It should be mentioned that zero-length routes have been used by this module to force a unit to stay in place but change

it's posture, e.g. Dig in for a defensive battle. The legal values for this are defined in the posturecodes database table, but current values are:

```
 posture |   posturetype
---------+-----------------
       1 | Admin. March
       2 | Engagement
       4 | Move to Contact
       6 | OffRoad Travel
       5 | Road Travel
       7 | Static Defense
       3 | Tactical March
       8 | Withdrawal
```

Note that the analyst is responsible for both deploying the assets in the posture for each group type and defining the defensive factor.

- Patrol – this is used for maritime patrol commands. It is the maritime patrol ID to invoke. Note that posture, routename and route ID are not used for maritime patrols. Just set this to 0 if you want a group to move.

The alternate_ato database table provides rules for when to activate an alternate ATO ruleset. In this case the ATO mission is defined the natural way, that is through the ATO GUI in scenario generation, and this table has only the following fields:

- scenario – The scenario within which this rule exists.
- check_id – The ID of the trigger
- mission – The ATO mission ID (as defined in the scenario generation GUI).

The final series of alternate plan responses relate to responses to a own-unit's health. As described in the trigger section, the trigger for this response is separate from the other triggers and is contained in the alternate_movement_2 and alternate_movement_2_list tables. The response is for one of more groups to move. Multiple groups can be given different commands based on a single trigger, for example commanding a weakened unit to hold for a limited time and then withdraw while telling multiple reserve units to step up and take the first unit's place. The database table defining the response is alternate_movement_routes. The table's fields are:

- Scenario – The scenario in which to use this rule.
- Plan – The plan ID. This must match the plan ID in the alternate_movement_2 table.
- Groupname, groupnumber – These are the group identifiers for the groups to be moved.
- Routename, routeid – These are the route identifier for the route to be taken by this group. Like the groupname and groupnumber, these are same as used throughout the scenario generation interface. So groups and routes should be set up in the scenario generator; only the alternate plan specifics currently need to be set up in SQL.

Database Prototyping Interfaces

None

Scenario Generation Interfaces

As described above, there's currently no presim GUI. The tables that need to be populated are fully described in the Algorithm section.

Runtime Interfaces

None

Applicable Data Analysis

While there are no data attributes tailored to track the invocation and success of alternate plans, some of the standard data collection options have been used to infer the success of alternate plans. These include:

- Losses. This is obvious.
- Kills by type. This can dramatically change with the invocation (or failure) of an alternate plan.
- Route points taken. While not commonly used for analysis, the system does maintain information on every route point achieved. This includes:
    - Notification of every route leg commanded or started in da_routeleg, and
    - Notification of the completion of the entire route, in da_route_completion. This can be used to measure success in attaining geographic locations.

  Additional route information is found in da_route_override. This table records all commander overrides of the original plan, which might be used to adjust the success in accomplishing objectives by not counting failure in completing routes if the route was overridden by the commander.