Tactical Planning in Space Game using Goal-Oriented Action Planning

Restuadi Studiawan[1], Mochamad Hariadi[1,2], Surya Sumpeno[1,2]

***Abstract*—Along with improvement of modern electronic games, necessity of an intelligent agent that easily build is needed. One of electronic games that need good intelligent agent is real-time tactics. In this game type, good action planning is necessary to provide best experience to the player.**
On this paper, we try to find out whether if Goal-Oriented Action Planning (GOAP) is robust enough to be used at tactical game. By using GOAP, tactic dynamism still can be provided with reasonable amount of runtime.
***Keywords*— Artificial intelligence, Games, Goal-Oriented Action Planning, Planning, Unity3D**

I. Introduction

Advancement of electronic game increase the necessity of challenging artificial intelligence. One way to achieve that is by creating artificial intelligence (AI) that can adapt its action based on observation and latest condition of game stage. Goal-Oriented Action Planning (GOAP) is one approach that offer adaptive capability to game artificial intelligence.

Since it being introduced at 2003 [1], some of attention given to GOAP to implement action of intelligent agent in electronic games [2][3]. GOAP is preferred to be used as agent with simple decision such as first person shooter (FPS) or action games but usage for agent with complex decision such as Real-Time Strategy (RTS) is less preferable due to its runtime speed and development complexity.

Here we find out whether if the complexity and runtime of GOAP is applicable in Real-Time Tactics game which have less complexity than RTS but have more decision to make rather than FPS games.

II. Related Works

To do this research, several method is reviewed.

A. Real Time Tactics

Tactical game is one genre that focused on composing set of actions needed to be done by units to achieve goal set by game scenario. There is two type of tactical game based on their time management on gameplay. When turn-based tactics give more time to think between phases by pausing the game after each action performed, real-time tactics (RTT) asks player to plan and think in real time to finish their mission. With that differences, AI built for each type of the games will have different approach. In real-time games it is necessary to make AI fast enough to comply with the nature of real-time gaming even though it process less calculation than AI on turn-based games.

[1] *Department of Electrical Engineering Faculty of Electrical Technology Institut Teknologi Sepuluh Nopember, Kampus ITS, Keputih, Surabaya 60111 INDONESIA*
[2, 3] *Department of Computer Engineering Faculty of Electrical Technology Institut Teknologi Sepuluh Nopember, Kampus ITS, Keputih, Surabaya 60111 INDONESIA*

RTT itself categorized as sub-genre of real-time strategy (RTS) games. Rather than composing unique actions for unit in game, RTS ask player to manage higher level strategy such as unit deployment and resource management. The line between RTT and RTS is kind of vague. This happened because games marketed using RTS as its genre sometimes have heavy emphasis on managing unit actions rather than the general strategy. In this research, definition used for describing tactical gaming is when game asks to finish missions with limited resource that fixed at beginning of scenario.

B. Goal-Oriented Action Planner

Jeff Orkin propose an approach of intelligent agent action planner [1] based on STRIPS [4] called Goal-Oriented Action Planner (GOAP). This approach is intended to build an intuitive and hospitable environment for Artificial Intelligent (AI) division that consist of people that comes from different disciplines.

GOAP is a technique for decision making that will produce chain of action called plan to reach a goal state that already set before. Core object in this technique is goals and actions. Goals is world condition that should be achieved by the agent, while action is an activity that can change world condition but have some conditions to be fulfilled before it can be used. A planner component will make a plan of actions so intelligent agent can use it to reach its goal.

Development of AI module shown to be simplified. Rather than creating a complex graph that need careful planning, developer can make nodes of action with visible input and output that can be evaluated in real-time as shown as in Fig 1.

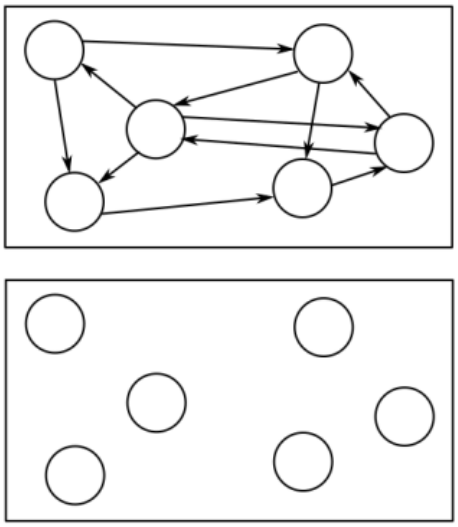

Fig 1: Simplification

Each action used in GOAP has precondition and effect which is a representation from world state. Precondition is the condition of world state that need to be fulfilled to do the action, and effect is the result of action that affect the world condition when it is running. Fig 2 shows the relation between objects in this technique.

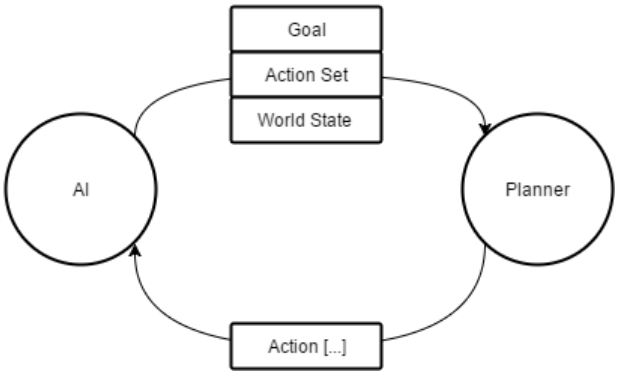

Fig 2: GOAP relation

Input for planner is current world condition, goal that want to be approached and set of action available for intelligent agent. Property of goal is part of world condition that will build a condition called goal world state. Plan formulated by doing search in state space graph to compose set of action with minimum value that will produce goal state from current condition. States other than goal state in approached condition will be ignored in this research.

C. Blackboard System

Blackboard system is an approach to manage data communication based on blackboard architecture model where a general knowledge base called blackboard is used and updated by many components in a system [5]. Characteristic of this system is freedom of agent expertise, discretion of problem-solving technique, flexible information representation, general interaction language, positioning metrics, event based activity, necessity of control and incremental solution generation. By using blackboard system, development of modules in game can focus on their function rather than how to communicate with other module as long as data standard is agreed at the beginning of development phase.

Fig 3 shows basic component from a blackboard system.

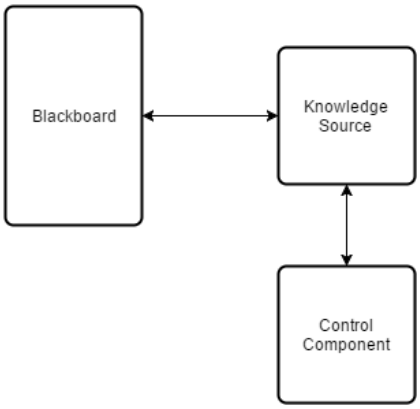

Fig 3: Blackboard System Component

D. Science Fiction Game Set in Space

The absence of real conflict in space make authors of science fiction makes condition that can be accepted by audience when the product is released. In example, Star Wars series (The Walt Disney Company) fighter use maneuver used by planes on World War II for artistic purpose. A lot of space based science fiction that follows use similar approach for their fighter maneuver even though there is no necessity to consider aerodynamics when moving in space. This decision usually taken because the audience used to see plane maneuvering inside atmosphere.

Unit organization usually use navy organization terms. The reason is because space fleet activity tend to have similar activity with its navy counterpart, such as necessity to maintain ship integrity while doing mission, planning of fleet movement and boarding procedure [6].

There is no difficulty to detect other vessel in space. For example, using available radio telescope on current days, we can find the location of Voyager I deep in space [7]. Based on this information, strategy and tactics in space based science fiction still having room for improvement.

Weapon is necessary to defeat hostile units. Fictional weaponry that usually used on space based science fiction can be categorized as follows.

1. Kinetic weapon, utilize its kinetic energy to deal damage. This type of weapon can launch with minimum technology complexity and can deal reasonable damage to its target. It have several shortfall such as the speed that not fast enough to reach far target and probability of collateral damage when used near settlement.
2. Energy based weapon, utilize non kinetic energy to deal damage. Weapon like laser beam and plasma launcher fall into this category. Its capability to reach target in high speed make this weapon preferable for engaging ranged target. But the weakness of this kind of weapon is the technology complexity, lower damage compared with kinetic weapon and the nature of dispersing at long range.
3. Smart weaponry, is a weapon having projectile that can adjust with its target movement. The speed of this kind of weapon is slow compared to other type, but can be loaded with other technology to improve its damage.

Interchanging between equipping and using type of weapon need tactical and strategic planning. There is some room to explore to find combination of equipment that can make space based games better.

III. IMPLEMENTATION

Game system used in this research was built using Unity3D engine. We use this game engine to utilize component based design paradigm that fit with modular trait of GOAP.

A. Game System

In this research, intelligent agent using GOAP architecture implemented to control team commander. Action used is order

to game units. Type of game unit that will be controlled by the intelligent agent described as follows.

1. Carrier Ship (CS), is a unit that have capability to store smaller units and change their equipment when inside. Other than that, every smaller unit health is restored when entering the carrier ship.
2. Variable Pod (VP), is unit with highest agility inside the game world. Its small size make it can be stored inside carrier ship. This unit have capability to change its equipment to do attack to appropriate target. Equipment that can be equipped to this unit is stored inside Carrier Ship. For this simulation, equipment that used can be seen as follows.
   - Micro Missile (M), have highest agility but its health is smallest. Used to do attack to another VP unit.
   - Cruise Missile (C), with its highest health, it can move into enemy defense line. Its attack value equal with micro missile, so it can be used to destroy enemy defense unit.
   - Torpedo (T), a missile with highest attack value. Its agility is lower than any other missile and have lower health than cruise missile. Used to deal damage to enemy carrier ship.
3. Energy Relay (ER). Is a unit that deployed to increase defense value from area around carrier ship. This unit equipped with laser that aim to hostile missiles entering its area. The laser damage is low and a cruise missile can go through it long enough. This way, energy relay is a target of ship equipped with cruise missile.

There is two type of action that can decrease game object health in this game. Both of it is missile and laser. A missile attack is an attack that have high damage value and used to inflict damage to available game units. Each missile have small amount of health. Laser is used to inflict damage to approaching hostile missile.

Game units is composed that way to make a rock-paper-scissor composition as seen at Fig 4.

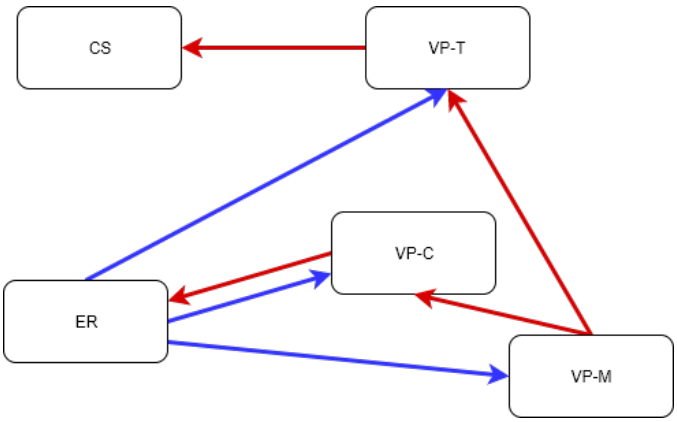

Fig 4: Game Units

Arrows that can be seen at Fig 4 is representation of unit effectiveness toward other type of unit. Red arrow means that unit can inflict damage to pointed type of unit, while blue arrow means that unit is good for blocking attack from pointed type of unit.

B. Game Flow

Basic flow of game composed on this research is observe-attack-reallocate. This flow is used to make agent adaptivity more visible to observe. At the beginning of gameplay, player can only observe status of their own fleet. Hostile fleet can be seen by player but their status is hidden before observed by any unit that they have.

To observe hostile unit, player should send a unit until the targeted unit entered its sensor range. Player can choose to attack observed target or reallocate unit equipment to match the property of observed hostile.

This flow can be restarted to observation state when change happened in hostile fleet when they send new unit to game stage.

C. Unit Logic

Logic used by units inside game use basic logic that implemented in accordance with its basic function. Basic logic of game units can be seen as follow.

1. Observation update.
2. Use laser to attack any hostile unit in range but prioritize hostile missile.
3. Seek nearest enemy if there is no special order.
4. If there is an order and the subject is hostile, attack targeted unit with missile when arrived at attack distance

Using that basic logic, each unit equipped with further logic to specify its role inside game. Logic specification of each type of unit can be seen as follow.

1. Carrier ship.
   - Launch available unit when asked.
2. Variable pod.
   - Return to carrier ship when missile is empty or unit health is low

D. Game System Interface and Group Control

Units created at this game have different action depend on its type. A standard to give order to unit is created so intelligent agents and player can call the same action. Three type of action is created for abstraction of control that can be used by player and intelligent agent.

Action that can be called described as follow.

1. Navigation. A basic action that can be used by any movable unit inside game. User of this action should describe action target before this action can be executed. Afterwards, approaching type should be selected. There is two type of movement that can be used by game object, which is:
   1. Approach target, game unit will approach designed target directly.
   2. Orbit target, game unit will orbit designed target.

   Distance from navigation target should be set after movement type selected. Unit will stop at that distance if selected movement is approach, or orbiting

navigation target with the distance as it radius when orbit target movement type is selected.
If navigation target is friendly carrier ship, unit will enter its hangar when arrived at certain distance.

2. Attack order. Is an action that owned by unit equipped with missile module. To use this action, user should provide attack target. After attack target provided, unit will send missile toward its target when the target is entering the missile attack range or directly ordered by action user using trigger method.
3. Hangar action. This action order hangar to equip a variable pod and launch it. User should provide what kind of equipment should be put onto variable pod before this action can be triggered.

E. Support Function

To help intelligent agent evaluate world condition, a standard of unit weighting is composed. Those functions can be seen as follows.

1) GetWeight

This function used to get game object weight compared to selected target. Weight produced by this function compare unit attack capability with target defense capability. Get weight function can be seen as follows.

$$w = \begin{cases} 0 & dpm - wf > 1 \\ 1 & dpm - wf \le 1 \end{cases} \times \begin{cases} (W_A \times atkDist) + dpm & pdu - wf > 1 \\ atkDist + (W_B \times dpm) & pdu - wf \le 1 \end{cases} \quad (1)$$

w : weight of game object compared to its target
dpm : *Damage Per Missile*, comparison between damage for each missile owned by unit and attack target
wf : basic factor of unit weight
W_A : weight constant for hostile unit that can be defeated using unit attack potential
W_B : weight constant for hostile unit that can't be defeated using unit attack potential
$atkDist$: attack distance
pdu : *Potential Damage Unit*, comparison between all attack value of selected unit and target health.

2) GetAttackDistance

This function calculate distance to attack. Comparison between maximum attack distance and optimal attack distance is the output of this function. Get attack distance notation can be seen as follows.

$$atkDist = min\{1, (mslHlt \div enDef) \times mslSpd \div rdrRng\} \quad (2)$$

$atkDist$: maximum attack distance that can be affirmed resulted with enemy health reduction
$mslHlt$: missile health
$enDef$: enemy defense value
$mslSpd$: missile speed
$rdrRng$: unit's radar range

3) FindAppropriateTarget

Function to find appropriate target. Its notation can be seen as follow.

$$target = max\left\{ w + \begin{cases} 3 & targetCheck = \text{VP-T} \\ 2 & targetCheck = \text{VP-M} \\ 1 & targetCheck = \text{VP-C} \end{cases} \right\} \quad (3)$$

$target$: selected hostile unit
w : game object weight
$targetCheck$: checked hostile unit

4) ArrangeHostile

Sorting enemy unit priority as target. Can be seen as follow.

1. Variable Pod, sorted by missile equipped, which is:
    1. Torpedo
    2. Cruise Missile
    3. Micro Missile
2. Energy Relay
3. Carrier Ship

F. GOAP Implementation

GOAP architecture created based on game design described on previous chapter. It is implemented to handle general tactics of intelligent agent inside gameplay. In this research, agent goal is to maximize friendly unit allocation to observed hostile unit. Scout will be sent when there is no hostile unit observed. Next planning state is to allocate existing stage unit and decision to add more unit or change equipments of variable pod that already sent on game stage.

1) General Strategy

Compare available units based on attack, defense and agility value. Action that can be done consist of:

1. Launch unit
2. Observe enemy
3. Allocate unit
4. Return unit

2) World State

World state is value that represents game world condition that will be evaluated to create action plan. This research use several world states that described as follows.

1. HostileUnobserved, represent total amount of unobserved hostile unit.
2. HostileObservedAssigned, represent total amount of observed hostile unit and there is at least one friendly unit allocated to that unit.
3. HostileObservedUnassigned, represent total amount of observed hostile unit that didn't allocated to any friendly unit.
4. FriendlyVPHangar, represent total amount of friendly unit that waiting inside carrier ship.
5. FriendlyVPStageAssigned, represent total amount of friendly unit exist on game stage and allocated to any other game unit.
6. FriendlyVPStageUnassigned, represent total amount of friendly unit on game stage and didn't allocated to any game unit.

3) Actions

Actions that used on GOAP purposed to maximize unit allocation toward hostile unit. Each friendly unit at game stage

will be evaluated and ordered to do action depend on each unit capability.

Action that used can be seen as follows.

1. Enemy Observation, is an action that pick one friendly unit to observe any unobserved hostile unit.
   - Precondition:
     1. Any unobserved hostile on stage.
     2. Any unassigned friendly VP.
   - Effect:
     1. No unobserved hostile unit on stage.
     2. Unassigned friendly VP decreased by one.
2. Allocate Unit, is an action that pick several units that did not have any target to be assigned to some action.
   1. Precondition:
      1. Any unassigned friendly VP on stage.
      2. Any unassigned observed hostile on stage.
   2. Effect:
      1. Unassigned friendly VP decreased as much as unassigned observed hostile on stage. Unassigned friendly VP amount will decrease to zero when unassigned observed hostile is more than its amount.
3. Deallocate Unit, is an action that deallocate any target of any unit that need to be directed to other target.
   - Precondition:
     1. Any assigned friendly VP on stage.
     2. Any hostile unit unassigned on stage.
     3. No VP on carrier ship.
     4. No unassigned friendly VP on stage.
   - Effect:
     1. Assigned friendly VP will decreased as much as unassigned observed hostile on stage. Assigned friendly VP amount will decrease to zero when unassigned observed hostile is more than its amount.
4. Launch Attack Unit, is an action to launch VP to stage. VP launched did not have a default target.
   - Precondition
     1. Any VP on friendly carrier ship's hangar.
     2. Any hostile on stage.
   - Effect
     1. VP on friendly carrier ship's hangar decreased as much as amount of hostile on stage or decreased to zero when hostile amount is more than available VP on hangar.
     2. Friendly VP on stage added as much as effect 1 value.
5. Launch Observer, launch unit to observe game world.
   - Precondition
     1. No VP unit on game stage.
     2. Any VP unit on hangar.
   - Effect
     1. VP on hangar reduced by one.
     2. Unassigned VP on stage added by one.

4) Planner

This research use GOAP planner that exist on github repository. This planner use A* to compose the plan and enough to accommodate implementation necessity.

Usage flow of the planner can be seen as follows.

1. Make Planner
   Actions used on GOAP declared here. Comparison function and nodes is declared here too.
2. Get Current World State
   After actions, comparison function and search function declared. Take current world condition to be the base of plan composition.
3. Decide Goal State
   On this section, goal state and planning declared. For this research, goal state that used is no hostile on game stage.
4. Run Planner to Get Plan
   Using goal state that described above, planning is executed on this section.

IV. Test and Analysis

A. Usage Test

World state defined by researcher to test whether the system approach in this paper can be run. To simplify observation of game unit when running simulation, some symbol is placed on every unit. Symbol used can be seen as follow.

1. ▉ Carrier Ship
2. ◊ Unobserved Ship
3. ▲ Variable Pod with Torpedo
4. ■ Variable Pod with Cruise Missile
5. ♦ Variable Pod with Micro Missile
6. ● Energy relay

Some scenario created to test the capability of intelligent agent that being built. This section will show the result of the test for each scenario. Every scenario tested for ten times and unit that counted on game stage is averaged to get unit count in game world.

1) Only Carrier Ship on Opponent Side

In this scenario, game will provide five unit of carrier ship. Intelligent agent is expected to launch VP unit with Torpedo as its equipment. Fig 5 shows the result from this test.

At the beginning of the scenario, GOAP agent sent a VP unit with cruise missile as its equipment. This is happening because the logic component decide to use it as the most appropriate unit to be launched when every hostile unit is unobserved.

After every hostile unit is observed, intelligent agent return the VP with cruise missile while launching all available unit with torpedo as its equipment. After that, all available unit attack hostile cruise ship until the session is over.

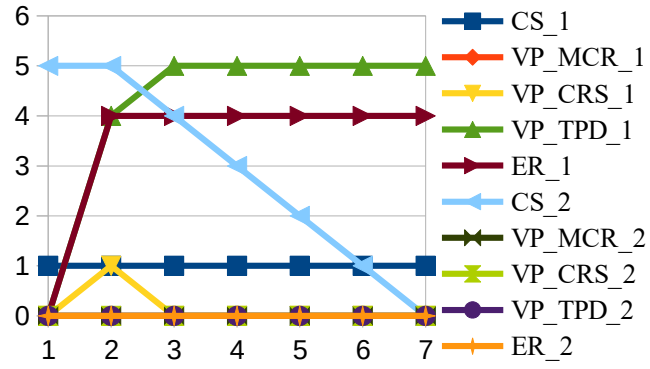

Fig 5: Against Five Carrier Ship

2) Only Defense Unit on Opponent Side

In this scenario, GOAP agent faced with ten defense unit. Expected behaviour is the agent send unit that brought cruise missile to destroy all hostile defense unit. Fig 6 show the result of this test.

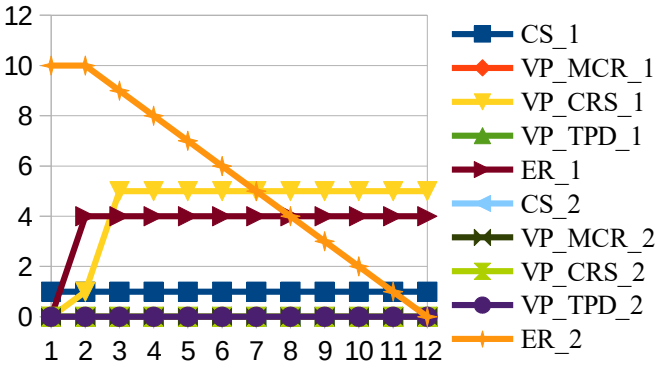

Fig 6: Against Ten Defense Unit

Agent sent one cruise missile unit to observe all hostile unit at the beginning of the test like previous test. After every hostile unit observed, intelligent agent decide to send remaining cruise missile unit to destroy all hostile defense unit.

3) Opponent Have Carrier Ship and Defense Unit

Opponent is having a carrier ship without hangar and four defense unit at this scenario. Intelligent agent is expected to eliminate every defense unit before destroying hostile carrier ship. Fig 7 show the result of this test.

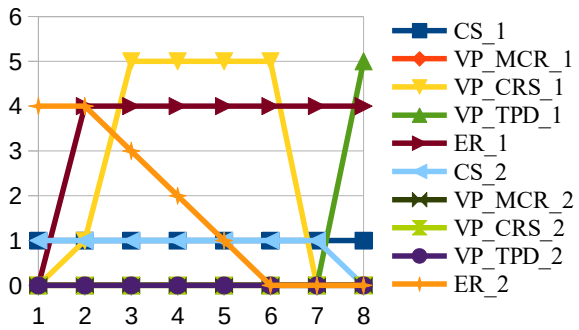

Fig 7: Against One Carrier Ship and Four Defense Unit

The first action is same with previous test, which is sending one cruise missile unit to observe every hostile unit. After that, more cruise missile unit sent out to eliminate every single hostile defense unit on stage.

When hostile defense unit already finished, agent call back all cruise missile unit and change their equipment to torpedo to destroy the remaining carrier ship.

4) Only Variable Pod on Opponent Side

At this scenario, there is five hostile variable pod unit on stage. Its composition can be seen as follow.

- Two unit equipped with torpedo.
- One unit equipped with cruise missile.
- One unit equipped with micro missile.

Expected behaviour from this scenario is the intelligent agent will eliminate hostile VP unit in this order. First, torpedo unit should be destroyed and then cruise missile. The last priority is VP unit equipped with micro missile. Fig 8 show the result of this test.

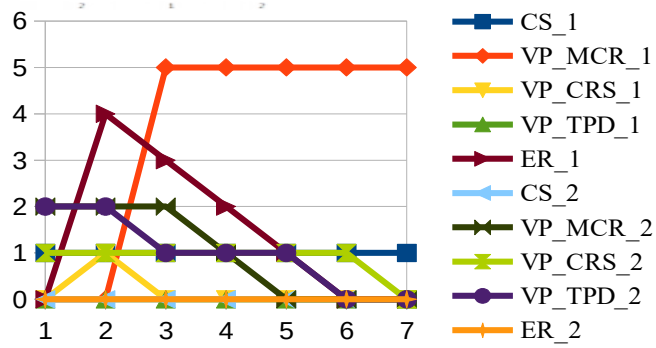

Fig 8: Against Five Variable Pod with Different Equipment

Result of this test is different from what is expected. Agent choose to eliminate units that equipped with micro missile first, then torpedo and lastly it order the attack units to eliminate remaining hostile unit equipped with cruise missile.

5) Opponent Have Carrier Ship and Variable Pods

At this test, there is one carrier ship without hangar and five VP unit equipped with weapons in same composition as previous section. Expected result is the agent eliminate all hostile VP in same order as previous section before start to destroy hostile carrier ship. Fig 9 show the result of this test.

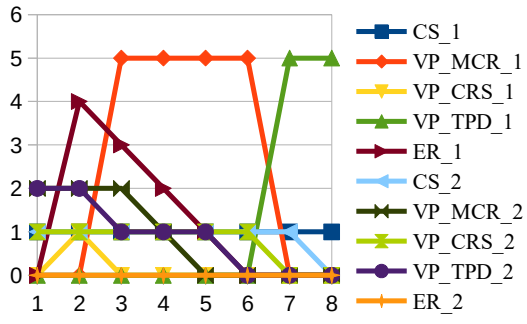

Fig 9: Against One Carrier Ship and Five Variable Pod with Different Equipment

Result of this test started with same sequence as the previous test does. After every hostile VP is eliminated, intelligent agent call back all micro missile unit and change their equipment to torpedo. Afterwards, those unit destroy hostile carrier ship.

6) Opponent Have Every Unit on Game Stage

All unit is placed in game world at this scenario. Their composition can be seen as follows.

- One carrier ship.
- Two VP with torpedo.
- One VP with cruise missile.
- Two VP with micro missile.
- Four defense unit.

Expected result from this scenario is the order of eliminating hostile started from hostile VP unit with same order as previous section, then defense unit need to be neutralized before moving to hostile carrier ship. Fig 10 show the result of this test.

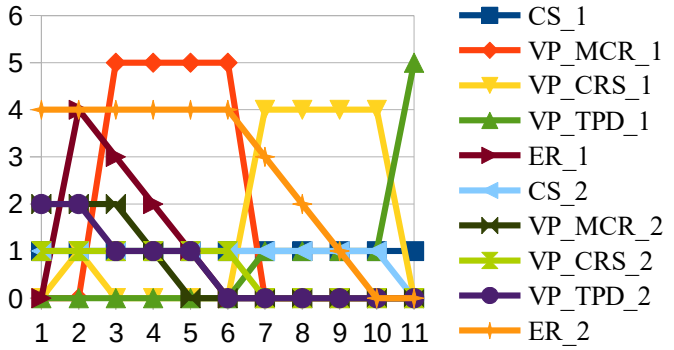

Fig 10: Against One Carrier Ship, Five Variable Pod and Four Defense Unit

This test shows that GOAP agent choose to eliminate hostile VP then move to hostile defense unit before trying to defeat carrier ship.

B. Speed

In general, planner make a plan with three to five nodes. Time used is below 0.01 second for each execution. This shows that GOAP is fast enough to be implemented at RTT games.

V. Conclusion and Future Works

From development that done in this research, GOAP helps to reduce complexity when designing intelligent agent. Using definition created in this research, composition dynamism of the units can be shown.

In other side, artificial intelligence designer should evaluate every action that have connection with a parameter when there is a change in parameter type. This is the difficulty when using GOAP.

Future works in this region can be done with providing more definition in gameplay tactics. Other than that, difficulty scaling can be implemented.

When execution time reduced by the amount of nodes created, plan composition can broken down again per unit action using hierarchy.

Reference

[1] J. Orkin, "Applying Goal-Oriented Action Planning to Games," *AI Game Program. Wisdom 2*, no. December, pp. 217–227, 2003.

[2] E. of A. D. Long, "Enhanced NPC Behaviour using Goal Oriented Action Planning," no. September, p. 110, 2007.

[3] P. Bjarnolf, "Threat Analysis Using Goal-Oriented Action Planning : Planning in the Light of Information Fusion," *Inf. FUsion*, p. 68, 2008.

[4] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artif. Intell.*, vol. 2, no. October, pp. 189–208, 1972.

[5] I. D. Craig, "Blackboard systems," *Artif. Intell. Rev.*, vol. 2, no. 2, pp. 103–118, 1988.

[6] M. Hipple, "Against An Air Force Space Corps: Space Belongs to the Navy!," 2017. [Online]. Available: http://nationalinterest.org/blog/the-buzz/against-air-force-space-corps-space-belongs-the-navy-21350. [Accessed: 15-Dec-2017].

[7] E. Landau, "Voyager Signal Spotted By Earth Radio Telescopes," 2013. [Online]. Available: https://www.nasa.gov/mission_pages/voyager/multimedia/pia17047.html. [Accessed: 02-Jul-2017].