



University of Maribor

Faculty of Electrical Engineering
and Computer Science



Adaptive Online Opponent Game Policy Modeling with Association Rule Mining

21st IEEE International Symposium on
Computational Intelligence and Informatics
(CINTI 2021)

Authors:

Damijan Novak (damijan.novak@um.si) and Iztok Fister Jr. (iztok.fister1@um.si)

19.11.2021

PRESENTATION AGENDA

1. Motivation and purpose of the article
2. Real-Time Strategy games
 - ☐ Main game components used in our work
 - ☐ microRTS game simulation environment
 - ☐ Real-Time Strategy game subdomains
3. Association Rule Mining
4. Proposed opponent game policy modeling method (ogpmARM)
 - ☐ Motivation, design and limitations of the proposed method
 - ☐ Concept of method
5. Experiment
6. Results and their interpretation
7. Future work

MOTIVATION

- ❖ Real-Time Strategy (RTS) games are difficult to master due to the high complexity of their game environments.
- ❖ In RTS games players have to take into account gameplay **aspects** such as production and resource management, gameplay decisions (strategical command, tactical decisions, micromanagement of units, and the low-level unit behavior), scouting, and sometimes even diplomacy, to **gain an edge advantage** in the game, which would result in victory for the player.
- ❖ For the player to get an edge advantage and defeat the opponent, constant overwatch must be made regarding what the opponent is doing (**opponent modeling**), with adjustments to their gameplay made accordingly.

PURPOSE

- ❖ We tackle the opponent modeling domain by utilizing the Machine Learning (ML) approach, specifically **Association Rule Mining** (ARM), to search for **rules** on the RTS data gathered during the gameplay.
- ❖ The focus is on obtaining results during the **online gameplay** (i.e. while the game is being played).
- ❖ The action part of the rules is matched to the **pattern**, where the patterns are represented with predefined **game policies**.
- ❖ By achieving a successful pattern match, a conclusion can be made regarding which game policy is used by the opponent, and proper adjustments and **countermeasures** can be executed.

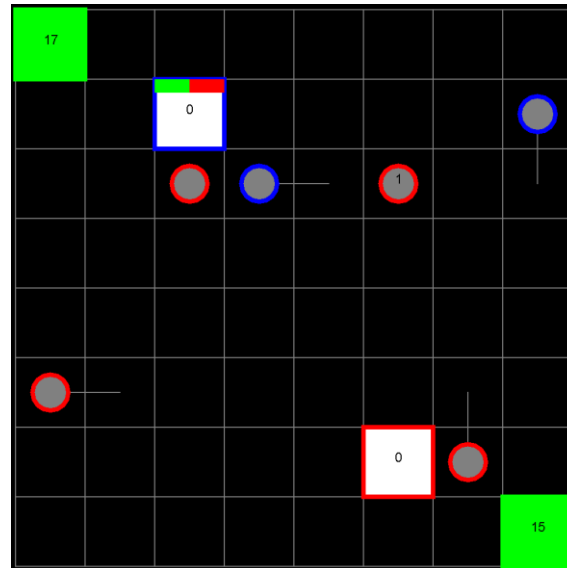
MAIN GAME COMPONENTS USED IN OUR WORK

We focused on four game components:

- ❖ **Feature**: a distinctive attribute or aspect of something. The commonly found features in RTS games are unit type, health of the unit, and terrain/map.
- ❖ **Replay**: a log of the already played game so far (i.e. the history of past states and the gameplay actions taken). We focus on specific **features** and past opponents' **actions** executed.
- ❖ **Rule**: defined as the regular continuous aspect of the game.
- ❖ **Pattern**: different recombinations of rules forming a pattern.

MICRORTS GAME SIMULATION ENVIRONMENT

microRTS is a small implementation of an RTS game, designed to perform AI research. It supports various sizes of game maps, multiple players, production of units and structures, resource gathering, durative actions (i.e. they are not instant and can last over multiple game states), options to make the game environment partially observable and non-deterministic, different battle game mechanics, etc.



microRTS

REAL-TIME STRATEGY GAME SUBDOMAINS: PURPOSE OF OPPONENT MODELING

- ❖ Playing the RTS game is more than just following a strategy plan set in a straight line.
- ❖ One must also be cautious about the **surrounding environment**, especially of every **action performed by the opponent**.
- ❖ If an assumption is made that an RTS game agent implements strategic and tactical command and can recognize opponents' intentions, a synergy of these two agents' mechanics should result in a better performance.
- ❖ Meaning, our agent can **change tactics** on the fly, and the **strategies** can be **altered** or **enhanced**.

REAL-TIME STRATEGY GAME SUBDOMAINS: GAME POLICY

Designing a general game policy for the RTS games is not an easy task due to the multiple simultaneous aspects of the game that must be considered.

- ❖ Game agents depend on a good design of game policy:
 - ❑ Must be robust enough to help deal with seemingly similar game situations.
 - ❑ Unpredictable, so that the opponent will have a hard time figuring out its plans.
 - ❑ Fast and dynamic in the sense of offering variability of operations in the same game scenarios.

ASSOCIATION RULE MINING

- ❖ Association Rule Mining (ARM) stands for an important ML method whose task is to find relationships between attributes/features in a transaction database.
- ❖ These relationships are presented as implications, where the left side of the mined rule represents an **antecedent** and the right side the **consequent**.

Example of the rule created by the microRTS features:

0.736111 (confidence value)

['enBases_1_1','enRanged_0_0','enResourcesLeft_20_20','enWorker_1_1','frBarracks_0_0','frBases_1_1','frResourcesLeft_20_20','primaryAction_4,']

==>

['baseThreatFriendly_false','enBarracks_0_0','enHeavy_0_0','enLight_0_0','frRanged_0_0','frWorker_1_1']

NUMERICAL ASSOCIATION RULE MINING

- ❖ Numerical Association Rule Mining (NARM) is considered as a variation of ARM.
- ❖ The main difference between NARM and ARM lies in handling numerical attributes, where, in NARM, **numerical attributes are handled without discretization**. It means that NARM algorithms operate directly with both categorical as well as numerical attributes.
- ❖ Discretization introduces certain problems which result in the loss of information and noise entering into data. Thus, NARM algorithms that do not need discretized data as an input provide more exact mined rules.
- ❖ Most of the NARM algorithms are **based on stochastic population-based nature-inspired algorithms** due to their ability to search the **vast search space**.

NUMERICAL ASSOCIATION RULE MINING (2)

- ❖ We used an uARMSolver (universal ARM Solver), a novel software tool for tackling the NARM.

<https://github.com/firefly-cpp/uARMSolver>


- ❖ uARMSolver covers all stages of the ARM process, from data preprocessing, rule mining, and visualization of results.
- ❖ Many NARM algorithms are included in this tool, e.g. algorithms based on **Differential Evolution** or Firefly algorithms.

PROPOSED METHOD: OPPONENT GAME POLICY MODELING WITH ARM

Motivation and design of the proposed ogpmARM method

- ❖ The main idea is to record every action the opponent has done in the game environment, along with all the relevant game feature information during the game playout.
- ❖ This information is saved because the past holds essential clues that can reveal the opponent's gameplay patterns.

record the opponent data  create a pattern upon the rules

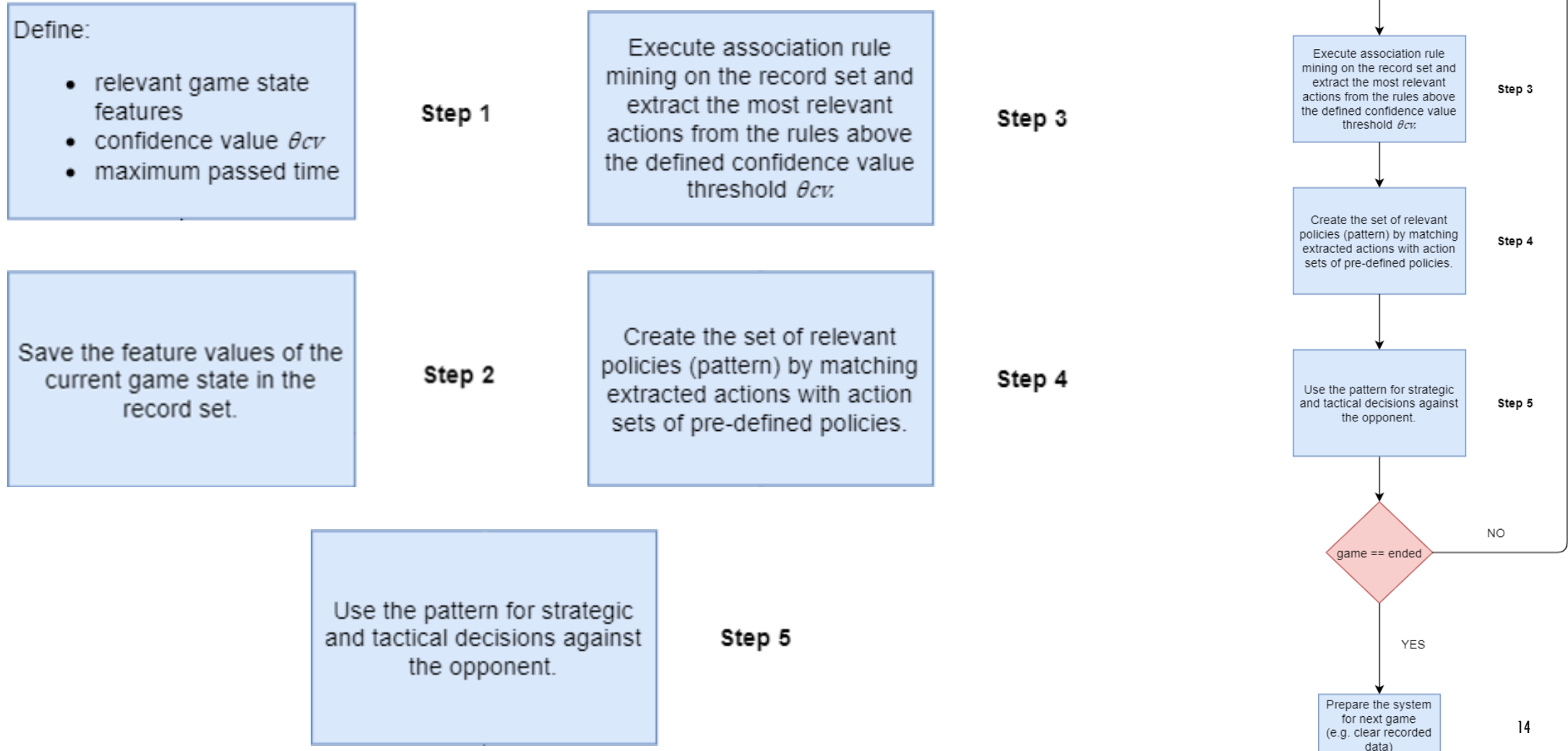
 find the most probable game policy that the opponent is using

PROPOSED METHOD: OPPONENT GAME POLICY MODELING WITH ARM (2)

Design and limitations of the proposed method

- ❖ Design is driven by choice to have a method capable of providing results without the forward model, i.e. **no simulations needed** into the possible future game states.
- ❖ The method is also designed to be **time adjustable** to enable the possibility of discovering what patterns the opponent is using in a chosen time frame.
- ❖ There are some limitations regarding this chosen design. The first obvious one is that if the opponent changes their gameplay patterns during the game, this will only be sensed in the following state(s), because **no future predictions** are made at this point.
- ❖ Automatic RTS feature identification and extraction are not implemented yet.

CONCEPT OF METHOD



CONCEPT OF METHOD (2)

❖ Step 1

Define:

- ❑ The relevant game features are recorded from the game state during the gameplay.
- ❑ The confidence value threshold θ_{cv} above which the rules provided by uARMSolver will be kept.
- ❑ The maximum passed time for which the rules are kept. It can also be expressed in the form of the value of passed game state iterations.

❖ Step 2

- ❑ Whenever the opponent executes the player-action, the values for chosen features from the game state are recorded, along with the action-type (e.g. move, attack, produce, and other actions).
- ❑ The record is only made during a set time interval.
- ❑ The closed time interval during which the records are kept is $[a - mpt, a]$, where a is the result of a function call `time(current_frame_num)`, and mpt is the maximum passed time for which the rules are kept.

CONCEPT OF METHOD (3)

❖ Step 3

- ❑ The NARM is executed upon the gathered data set.
- ❑ Rules above the θ_{cv} are selected.
- ❑ The action-types are extracted from the rules. That action-type represents the action that has the highest confidence for being the primary action type that the opponent is using.

❖ Step 4

- ❑ The policies where the extracted action type is present are selected from the pre-defined set of game policies.
- ❑ The set of chosen game policies forms a pattern.

❖ Step 5

- ❑ Use the pattern for strategic and tactical decisions against the opponent.

EXPERIMENT

- ❖ microRTS game simulation environment was used.
- ❖ The following relevant game state **features** were defined for which the values will be recorded: Number of friendly light, heavy and ranged units, number of friendly workers, number of opponent light, heavy and ranged units, number of opponent workers, flag if a friendly base is under threat (true / false), number of friendly and opponent resources left, number of friendly and opponent bases, number of friendly and opponent barracks.
- ❖ The following possible **actions** from which to choose are first defined: No action taken (0), move (1), harvest (2), unit returns (3), produce (4), attack location (5).

EXPERIMENT (2)

Three basic policy action sets are created from the set of possible actions:

- ❑ Resource gathering/harvesting policy:

No action taken (0), harvest (2).

- ❑ Tactical policy:

No action taken (0), move (1), unit returns (3), attack location (5).

- ❑ Production policy:

No action taken (0), produce (4).

EXPERIMENT (3)

❖ Experiment settings:

- ❑ The confidence value θ_{cv} was set to 0,5, and the maximum passed time was set to -1 (so that all the records from the first game frame forward are kept).
- ❑ The maximum running game time was set to 3,000 frames.
- ❑ The uARMSolver and microRTS experimental environments were used with default values.

❖ Experiment design

- ❑ The experiment included three gameplaying agents: RandomBiasedAI, UCT, and NaiveMCTS.
- ❑ The testing game agents played against the RandomAI agent on the pre-included microRTS map called basesWorkers16x16.
- ❑ Each of the game agents played the game three times.

RESULTS

Note: The numbers inside each action set are indexed by the action index, and they show the intensity of the action being used.

	RandomBiasedAI	UCT	NaiveMCTS
1.	500 th : {5, 25, 8, 4, 8, 0}	500 th : {0, 28, 7, 5, 5, 0}	500 th : {7, 27, 6, 6, 3, 0}
	1,000 th : {7, 30, 6, 5, 5, 0}	1,000 th : {10, 18, 5, 7, 3, 3}	1,000 th : {9, 19, 9, 5, 0, 4}
	1,500 th : {8, 16, 3, 4, 6, 1}	1,500 th : {6, 17, 9, 5, 3, 2}	1,500 th : {10, 21, 8, 3, 3, 2}
	2,000 th : {3, 16, 8, 5, 6, 4}	2,000 th : {0, 29, 4, 8, 3, 0}	/
	2,500 th : {4, 27, 5, 4, 9, 1}	/	/
	3,000 th : {4, 17, 9, 5, 6, 2}	/	/
2.	500 th : {16, 16, 5, 8, 12, 0}	500 th : {4, 19, 9, 7, 4, 0}	500 th : {5, 25, 7, 6, 8, 0}
	1,000 th : {7, 10, 4, 4, 4, 0}	1,000 th : {5, 18, 5, 6, 2, 4}	1,000 th : {8, 23, 7, 4, 5, 0}
	1,500 th : {3, 21, 3, 3, 6, 4}	1,500 th : {7, 15, 6, 5, 2, 1}	1,500 th : {10, 21, 6, 3, 5, 6}
	2,000 th : {9, 11, 3, 5, 7, 1}	2,000 th : {8, 23, 5, 9, 1, 4}	/
	2,500 th : {5, 17, 3, 2, 5, 3}	2,500 th : {6, 28, 8, 6, 1, 3}	/
	3,000 th : {10, 16, 4, 5, 7, 2}	/	/
3.	500 th : {17, 12, 3, 6, 7, 0}	500 th : {12, 25, 8, 8, 2, 0}	500 th : {7, 18, 6, 5, 1, 0}
	1,000 th : {19, 14, 5, 5, 8, 0}	1,000 th : {5, 20, 6, 5, 2, 1}	1,000 th : {5, 19, 4, 5, 6, 4}
	1,500 th : {15, 18, 3, 7, 5, 0}	1,500 th : {4, 16, 4, 6, 3, 5}	1,500 th : {6, 18, 7, 5, 5, 3}
	2,000 th : {9, 12, 3, 9, 2, 2}	2,000 th : {7, 18, 9, 4, 2, 2}	2,000 th : {8, 23, 3, 4, 4, 1}
	2,500 th : {13, 14, 3, 5, 7, 4}	/	/
	3,000 th : {7, 15, 6, 7, 4, 2}	/	/

RESULTS INTERPRETATION

- ❖ From the Table, it can be observed that all of the indexed actions have appeared in it, and, therefore, the patterns of all the basic policies are present. Meaning, all the game agents are capable of playing full-blown RTS game.
- ❖ There is a heavy presence of the action move (1) across all three agents and all the frames` measures. When combined with the action's unit returns (3), and attack location (5), the **tactical (policy) pattern** is formed (step 4), which is an important opponent behavior to look out and care for. This pattern never developed in the 500th frame (i.e. we did not have agents utilizing early rush/attack tactics). It only starts appearing in the 1,000th frame (and only UCT always utilized it until then).
- ❖ Agents who use tactical pattern early on are clearly offensive oriented. Step 5 of the ogpmARM would therefore create countermeasures like additional unit production and increased tactical policy usage.

RESULTS INTERPRETATION (2)

NaiveMCTS data reveal interesting facts:

- ❑ In all three games, at the beginning of the game, the agent utilizes a fair amount of resource gathering pattern (harvesting actions (2) for the 500th frame are: 1st game - 6x, 2nd game - 7x, 3rd game - 6x).
- ❑ In the 1st game the intensity of **the production pattern** is low (action produce(4)), but **the tactical policy pattern** is present.
- ❑ In the 2nd and 3rd games, the agent utilizes a fair amount of the production pattern, as well as incorporation of a tactical policy pattern, and both stay consistent across the middle and/or end games.

FUTURE WORK

- ❖ To test the method in other game environments (e.g StarCraft / SparCraft) and RTS game sub-genres (e.g. tactical RTS games – time analysis).
- ❖ Automatic feature selection / extraction / generation.
- ❖ Automatic policy creation based on patterns.
- ❖ Comparing the method to the different RL methods (e.g. the Proximal Policy Optimization algorithms).

Thank you for your time. Please join the discussion.