Modeling Large-Scale Warfighter Cognitive Reasoning and Decision Making Using Machine Learning (ML), Artificial Intelligence(AI), and Game Theory (GT)

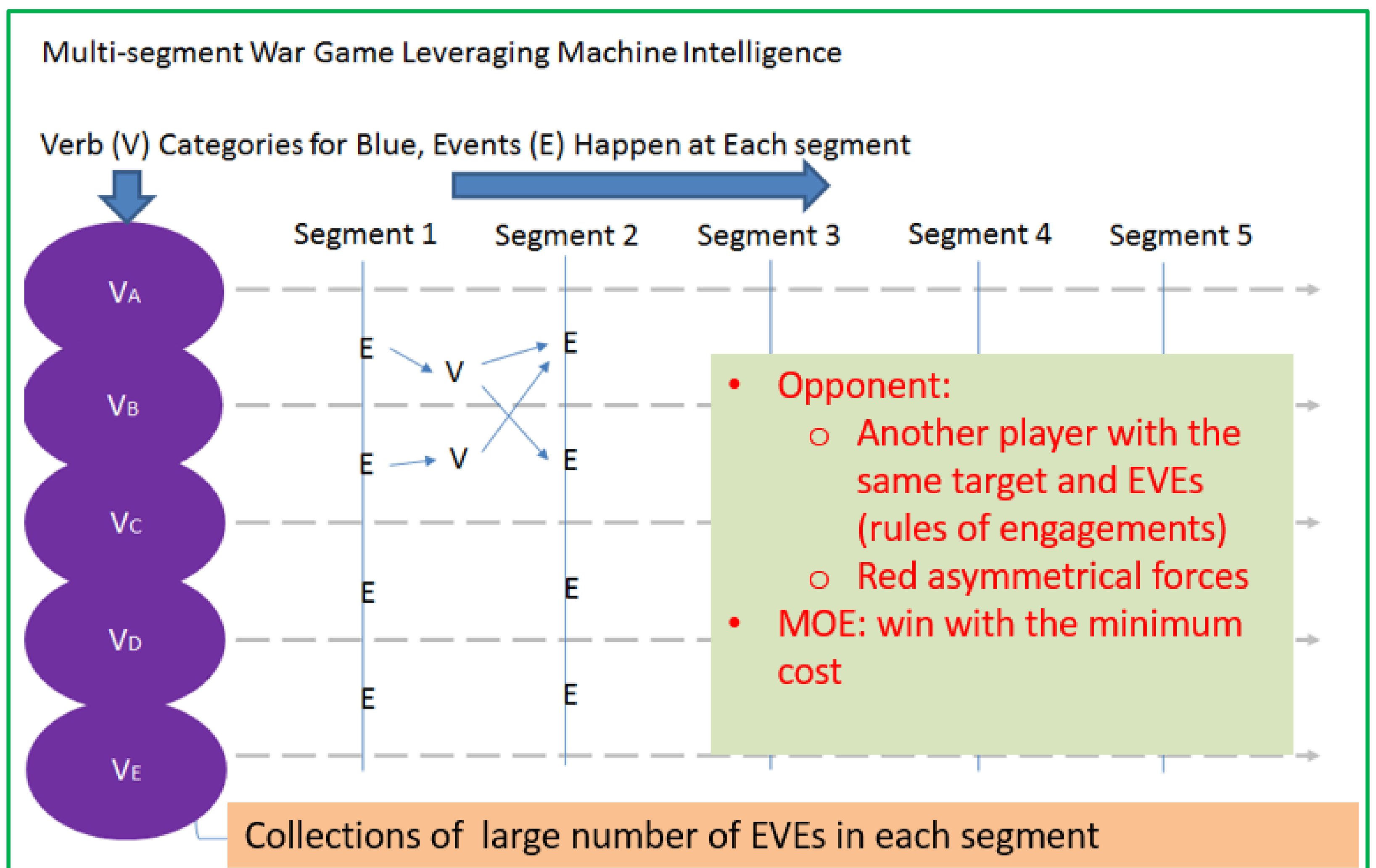
Ying Zhao, Ph.D., Naval Postgraduate School, Monterey, CA, USA, yzhao@nps.edu

10/2019

Naval Postgraduate School

Big Picture, Challenges, and Goals

- Cognitive architecture, algorithms, and software systems are important to model complex reasoning, cognitive functions, and decision-making in warfighting environments.
- Apply the ML/AI/GT techniques such as modeling, simulation, and readiness calculation to military applications to achieve decision-making superiority in the vast, complex, and uncertain areas of Cybersecurity and Information Warfare, including such applications as combat identification, Battlespace Awareness, C-C4ISR, Assured C2, modeling/simulation, and mission planning and war games.



Research Data Set and Tools

Table 1: Sequential or Parallel Asymmetric Action Combinations

Self-player	Opponent (e.g., environment or adversaries)
Action/state combination d_1	o_1
Action/state combination d_i	o_i
…	
Action combination d_N	o_M

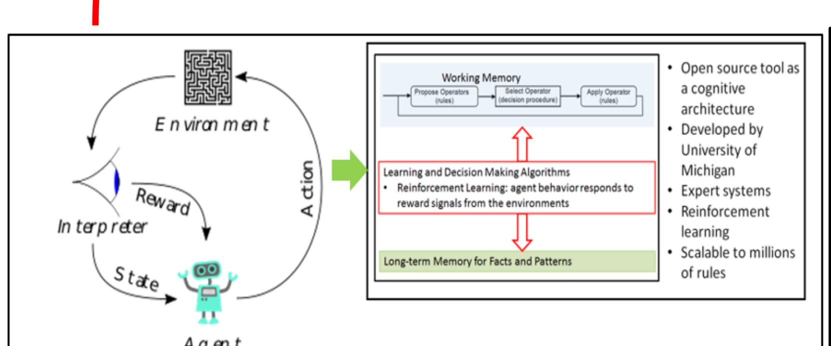
Table 2: Action Combination Components

Action/state combination component	c_1	c_i	…	c_k	End reward
	1	0		0	win
	1	1	0	1	not win

states or actions

Representation: Boolean lattice including counterfactuals

Soar-RL large-scale test data: 1.3M training combinations/400K test combinations and 50 attributes, ~25 attributes are state variables, and ~25 are action variables



Generic Reinforcement Learning

Soar-RL

Q (value): expected value to win the final game

- AlphaZero Q=f[state(t),action(t)]: state is an image, action is a valid move.
- Soar $Q(s',O')=Q(s,O)+\alpha[r+\gamma Q(s',O')-Q(s,O)]$: r is the reward for a combination of state and actions
- …

Attacker and Defender Dynamics and Coevolution Framework/Algorithm

MIT CSAIL: adversarial_dynamics_position.pdf

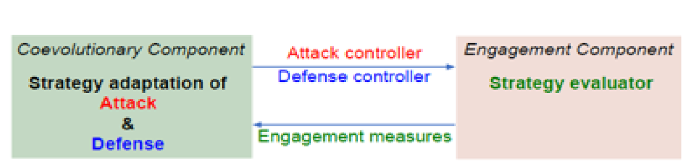


Figure 1: Component overview of our coevolutionary adversarial AI framework. The coevolutionary component performs search over the adversary controllers. The engagement component evaluates the strategies of the adversaries and returns the measurements of the engagement.

Conclusion and Future Work

- Leveraged and previous research results into different applications
- Explored new ML/AI algorithms in complex systems
- Future work: test Soar-RL, Coevolution, and LAILOW in real-life exercises, war games, and warfighter AI assistant implementations

Research Methods

- Leverage AI to Learn, Optimize, and Win for a complex enterprise

LAILOW process: Leverage AI to learn, optimize, and win for a complex enterprise

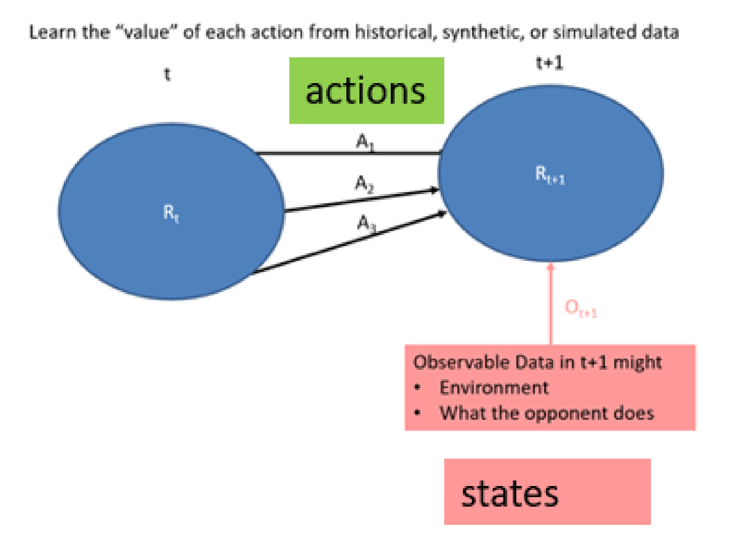


Figure 2. LAILOW learns the value of each action/state combination.

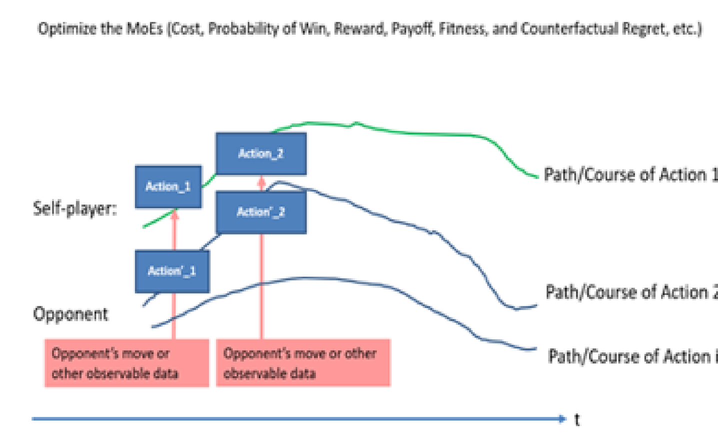


Figure 3. The "Optimize" Step

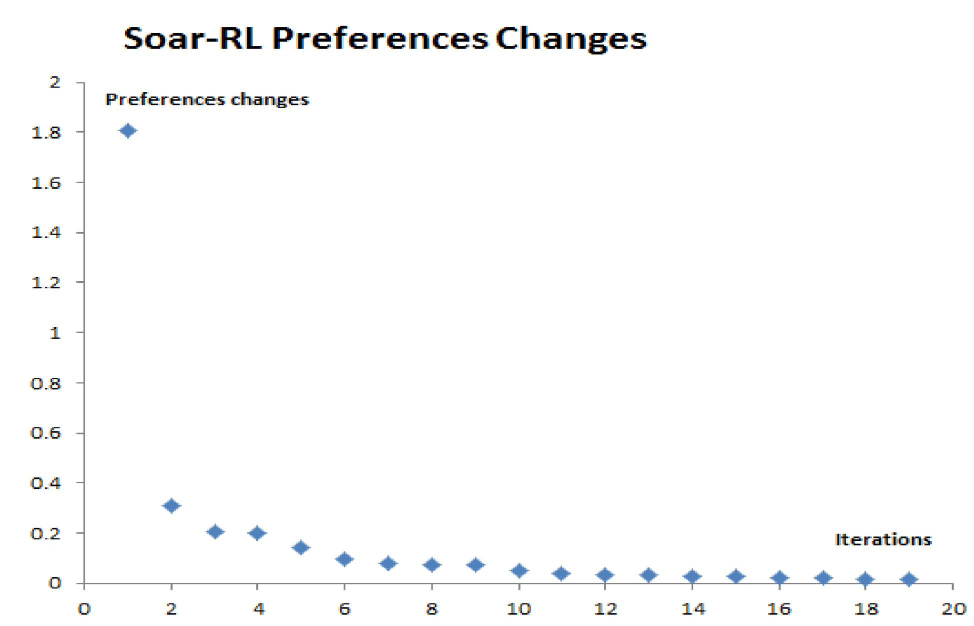
Actions: Self-player can control and decide.

States: Data that are not control and choose by the self-player but can affect the end result such as opponent's move, environment, etc. (States can be imperfect/hidden – need to learn/approximate/predict)

Unclassified

Research Results

- Showed Soar-RL learning, adaptation, and convergence for the big data set



"Causal Learning in Modeling Multi-segment War Game Leveraging Machine Intelligence with EVE Structures." Paper accepted to the AAAI 2019 Fall Symposium.

- Advanced Soar-RL successfully to the CEC Combat ID; will be tested in the Trident Warrior 2020 exercise
- Integrated Soar-RL with the coevolution framework in a war game in the context of over-the-horizon targeting

Self-play with a fixed opponent

```
population_size: 100
max_length: 10
elite_size: 1
generations: 3
tournament_size: 2
seed: 1
crossover_probability: 0.8
mutation_probability: 0.1
codon_size: 127
integer_input_element_max: 1000
bnf_grammar: "tests/grammars/iterated_prisoners_dilemma_attacker.bnf"
fitness_function:
    name: "IteratedPrisonersDilemma"
    opponent: "lambda h, i: 'NO'"
    n_iterations: 20
```

Coevolution Configuration (Coev.yml)

```
populations:
  attacker:
    adversary: defender
    bnf_grammar: "tests/grammars/iterated_prisoners_dilemma_attacker.bnf"
    fitness_function:
      name: "IteratedPrisonersDilemma"
      opponent: None
      n_iterations: 20
  defender:
    adversary: attacker
    bnf_grammar: "tests/grammars/iterated_prisoners_dilemma_defender.bnf"
    fitness_function:
      name: "IteratedPrisonersDilemma"
      opponent: None
      n_iterations: 20
```

Unclassified

1. configurations\iterated_prisoners_dilemma.yml

```
population_size: 100
max_length: 10
elite_size: 1
generations: 3
tournament_size: 2
seed: 1
crossover_probability: 0.8
mutation_probability: 0.1
codon_size: 127
integer_input_element_max: 1000
bnf_grammar: "tests/grammars/iterated_prisoners_dilemma_attacker.bnf"
fitness_function:
    name: "IteratedPrisonersDilemma"
    opponent: "lambda h, i: 'NO'"
    n_iterations: 20
```

2. grammars\iterated_prisoners_dilemma_attacker.bnf

```
<strategy> ::= lambda h, i: <eves>
<eves> ::= "<attribute1><attribute2><attribute3>"
<attribute1> ::= 1|0
<attribute2> ::= 1|0
<attribute3> ::= 1|0
```

3. Payoff lookup (m&s data): game_theory_game.py

```
EVE1: str = "001"
EVE2: str = "100"
EVE3: str = "010"
EVE4: str = "111"
EVE5: str = "011"
EVE6: str = "110"
EVE7: str = "000"
EVE8: str = "101"
NO: str = "NO"

PAYOFF: Dict[Tuple[str, str], Tuple[float, float]] = {
    (EVE1, NO): (1.7, 0),
    (EVE2, NO): (1.5, 0),
    (EVE3, NO): (0.8, 0),
    (EVE4, NO): (2.4, 0),
    (EVE5, NO): (0.9, 0),
```

4. Fitness

```
@staticmethod
def get_fitness(sentences: List[Tuple[float, float]]) -> float:
    """ Fitness is the negated sum of the sentences
    """
    fitness: float = -sum([_[0] for _ in sentences])
    return fitness
```


Researchers: Dr. Ying Zhao (PI), yzhao@nps.edu

Sponsor: Naval Postgraduate School Research Program (NPS-19-N091-A)

Topic Sponsor: N81I0, Information Warfare Branch