An Artificially Intelligent Battleship Player Utilizing Adaptive Firing and Placement Strategies

PENNSTATE 1855

The **Computer Science** and **Engineering Department**

Jeremy G. Bridon, Zachary A. Correll, Craig R. Dubler, Zachary K. Gotsch

Dr. Steven Shaffer; Advisor – The Pennsylvania State University -- *Research for CMPSC 422; Artificial Intelligence*

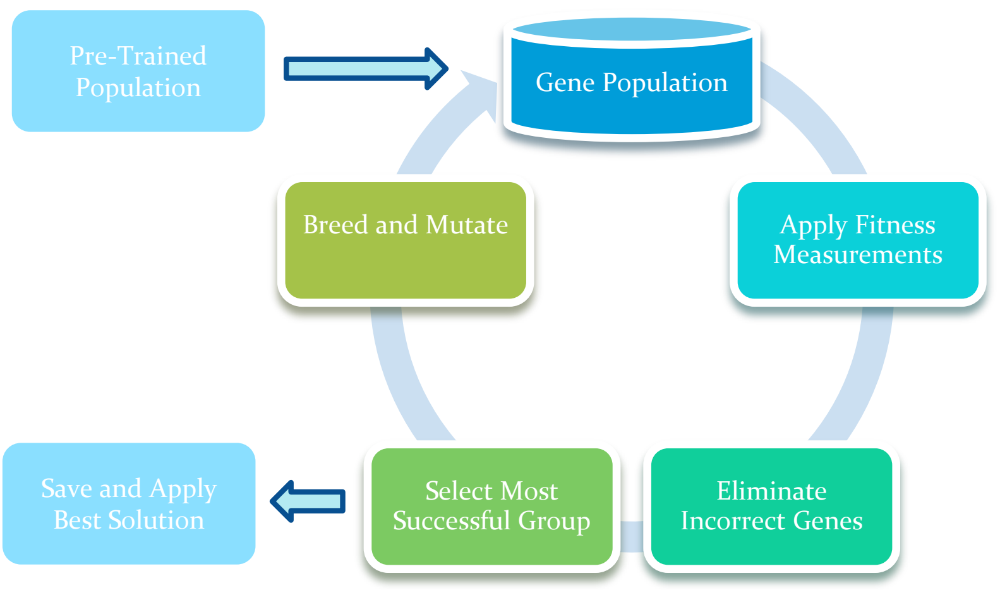
Research

Abstract & Introduction

- Research, design, and analyze an **Artificially Intelligent** (AI) **player** for the **classic game of Battleship**
- Apply AI methods to **ship placement, targeting**, and **sinking strategy**
- Test solution in **competitions** against other **artificially intelligent** battleship opponents
- There is no finite, or algorithmic, solution due to:
  - The unknown opponent logic
  - The enormous complexity in the solution space

Methodology

- Common Artificial Intelligence approaches:
  - Rule-based and Expert Systems
    - Follow a series of hand-coded rules that the computer will follow, attempting to apply high-level logic on it's own
  - Neural Networks
    - Create, reinforce, weaken, or destroy connections between solution representations
  - Genetic Algorithms
    - Breed, mutate, and measure solution representations, treated as genes


Pre-Trained Population
Gene Population
Apply Fitness Measurements
Eliminate Incorrect Genes
Select Most Successful Group
Save and Apply Best Solution
Breed and Mutate

Genetic Algorithm Cycle ←
Genetic algorithms take the concept of the evolutionary theory and apply it to a computationally expensive problem; analogous to a "search algorithm" but with heuristic properties.

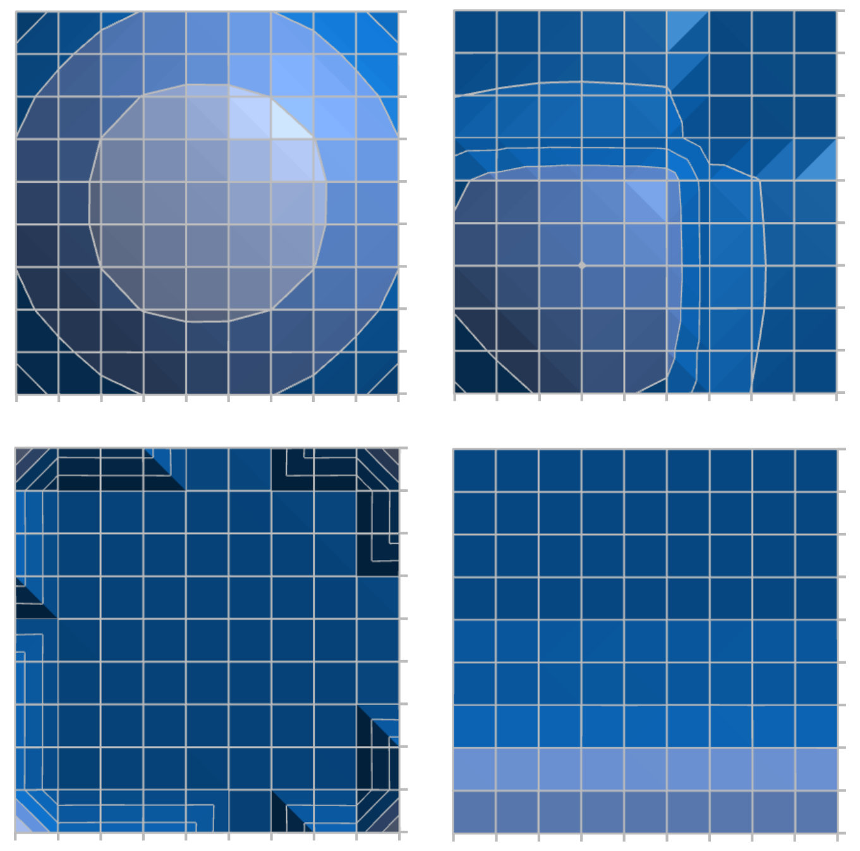
Genetic Algorithms

- A class of heuristic algorithms:
  - An application of evolutionary algorithms
  - Provides acceptable solutions but lack a formal proof
- Similar to the evolutionary theory:
  - Gene – Solution
  - Gene population – Solution set
  - Selection – Accuracy of solution
  - Reproduction – Convergence of possible solution
  - Termination – Removal of unfit solutions
  - Mutations – Introduction of new solutions

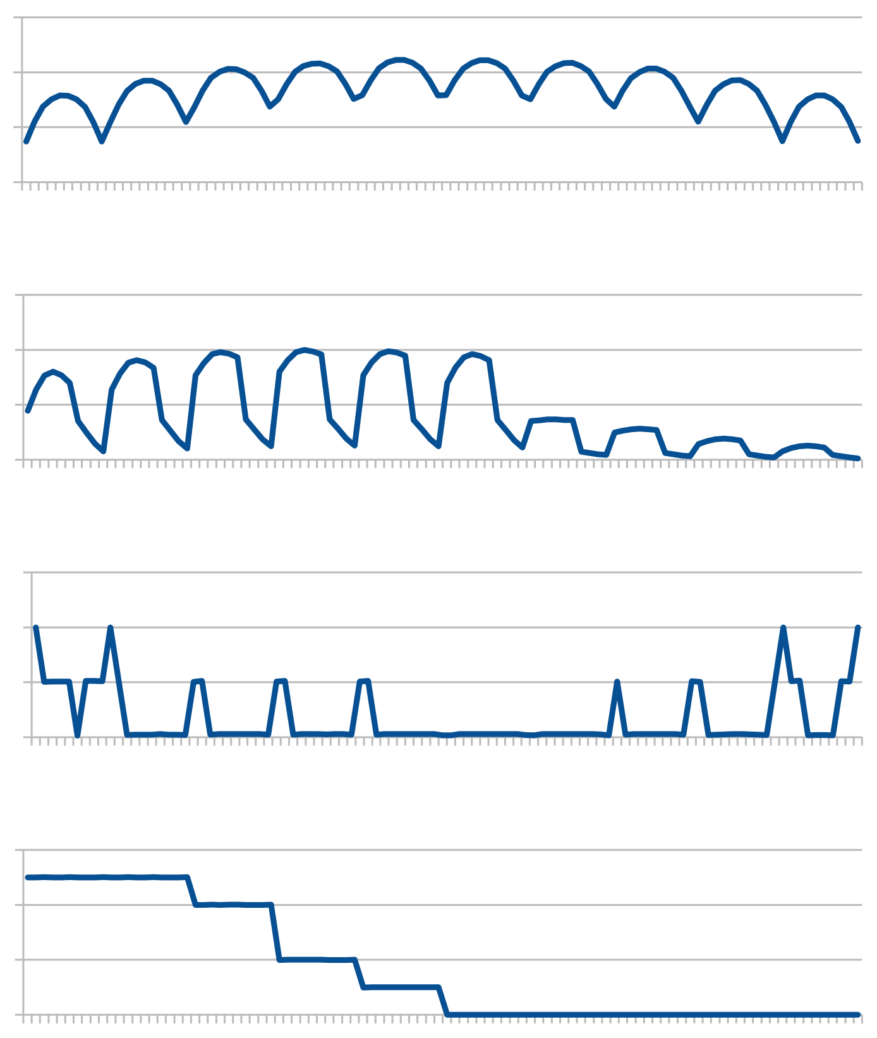
AI Design

Solution Representation

- Problem broke down to **three components** that have different AI approaches for solution searching:
  - **Ship placement** – Sequential Monte Carlo Method enemy shot pattern recognition
    - Save all enemy shots and density maps over time, apply Monte Carlo algorithm, and place ship to safety (See below graphs)
  - **Targeting** – Genetic algorithm to track enemy ship placement via pattern recognition
  - **Sinking strategy** – Genetic programming to optimized sinking logic



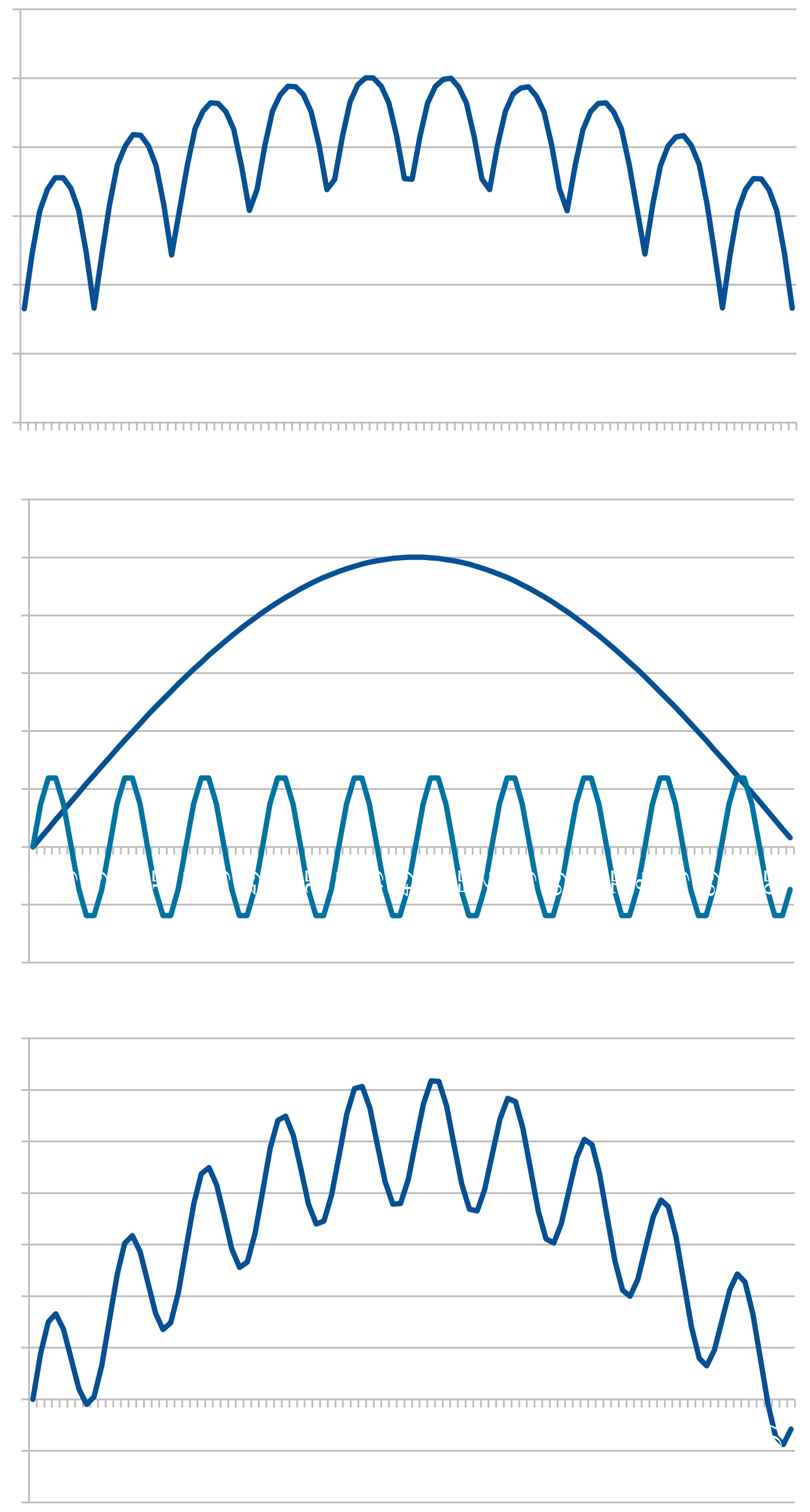
Placement Density Maps ↑
Different methods of ship placement over 1,000,000 rounds. Lighter color indicates higher density of ship placement. From top left, clockwise: Random, lower-left corner favored, bottom wall, one ship in each corner.



Placement Density Linear Functions ↑
Different methods of ship placement over 1,000,000 rounds; graphed as a linear function. From top to bottom: Random, lower-left corner favored, one ship in each corner, bottom wall. Same as the left maps.

Ship Targeting & Shooting

- Uses a genetic algorithm for quick adaptation to enemy ship placement patterns
  - Even with random placement (See left graph) a pattern does emerge
- Any enemy placement algorithm will generate a harmonic function
- All harmonic functions can be decomposed into a combination of sine and cosine wave that can be then used to generate a probability function
- We define a gene as five harmonics that are randomly chosen at first
- This harmonic represents a probability distribution of the likely hood of enemy ship placement
- Gene-mutation and cross-over will converge the gene to a close-fit solution
- Gene fitness, or validation of a possible solution, is done through a Fast Fourier Transformation



Sample Target Data and Gene Harmonics ←
To the left, the top-most graph is the linear function of the random ship placement density map. Underneath that is a harmonic gene with only two components. With these two components, there is enough data to simulate the target pattern found in the initial density map, which is drawn on the bottom-most graph.

Ship Sinking Logic

- Sinking is based off of genetic programming, the application of genetic algorithms to an instruction set rather than data
- This is needed since dynamic instructions are needed to sink a ship
- A state machine is also needed due to the complexity of sinking a ship: Targeting, locking, sinking
- Over 15 assembly-like instructions were created that include conditionals, target movement, shooting, etc..
- Each gene is cut into three "states" with 20 instructions each except for the first which is reduced for simplicity
- The initial gene pool is pre-trained against hard-coded logic

Outcome

Implementation & Analysis

- Written in C/C++ for WIN32/UNIX/Unix-Like platforms
- Open-Source at ***http://code.google.com/p/battlestar-ai/***
- Competed against several hand-coded opponents; Successful in all aspects of placement, shooting, and sinking
- Observations:
  - In a same-skill matches, the player to shoot first had a 10% greater chance of winning
  - Checker-board patterning greatly improved ship-hit rate to 70% compared to full-board shooting
  - AI ship placement successfully avoided random shooting and avoided intelligent learning methods
  - Targeting & shooting performed well at pattern patching
  - Ship sinking logic outperformed hard-coded solutions that lacked learning-sinking logic on average by 90%

Conclusion

- The use of genetic algorithms is both a **valid** approach to AI and an **effective** way at learning, pattern matching, and searching
- Genetic algorithms are especially promising for large-scale future applications due to the growth of multi-core processors; helpful in splitting fitness and breeding processing

References

[1] K. Benson, "Evolving Finite State Machines with Embedded Genetic Programming for Automatic Target Detection". Congress on Evolutionary Computation, 2000, vol. 2, pp. 1543-1549. Jul, 2000.

[2] S. M. Cheang; K. Sak; K. H. Lee, "Evolutionary parallel programming: design and implementation". Evolutionary Computation, vol. 14, n.2, pp. 129-156. Jun. 2006.

[3] F. Dellaert; D. Fox; W. Burgard; S. Thrun, "Monte Carlo Localization for Mobile Robots". IEEE International Conference on Robotics and Automation (ICRA99). May, 1999.

[4] IEEE Intelligent Systems staff. "Genetic Programming". IEEE Intelligent Systems, vol. 15 n. 3, pp.74-84, May 2000.

[5] Q. C. Meng; T. J. Feng; Z. Chen; C. J. Zhou; J. H. Bo, "Genetic Algorithms Encoding Study and A Sufficient Convergence Condition of GAs". Systems, Man, and Cybernetics, 1999. IEEE SMC '99, vol. 1, pp. 12-15, Oct. 1999.

[6] N. Metropolis; S. Ulam, "The Monte Carl Method". Journal of the American Statistical Association, vol. 44, pp. 335-341, Sep. 1949.

[7] D. Monniaux, "An Abstract Monte-Carlo Method for the Analysis of Probabilistic Programs". ACM SIGPLAN Notices, vol 36, pp. 93-101. Mar. 2001.

[8] M. Mysinger, "Genetic Design of an Artificial Intelligence to Play the Classic Game of Battleship". Genetic algorithms and genetic programming and Stanford, pp. 101-110. 1998.

[9] A. Roy, "Artificial Neural Networks - A Science in Trouble". 2000 ACM SIGKDD, vol. 1, n. 2, pp. 33-38, Jan. 2000.

[10] T. Weise; M. Zapf; K. Geihs, "Rule-based Genetic Programming". Bio-Inspired Models of Network, Information and Computer Systems, pp. 8-15, Dec. 2007.