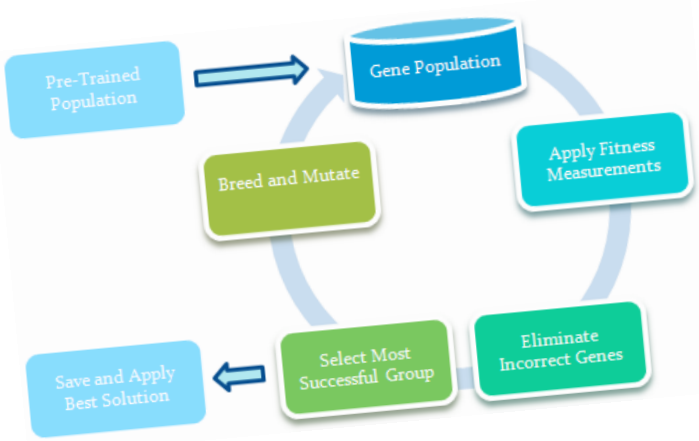
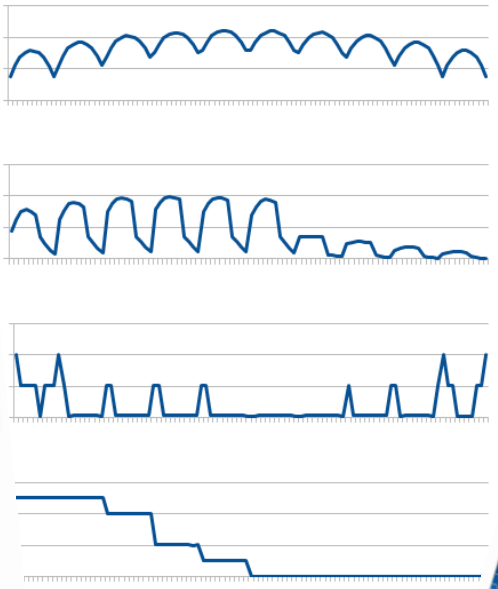
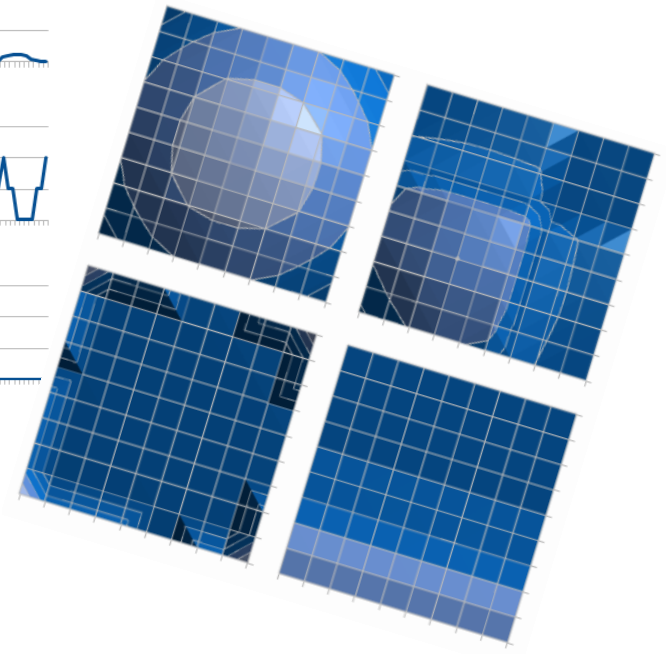
An Artificially Intelligent Battleship Player Utilizing Adaptive Firing and Placement Strategies

Jeremy G. Bridon, Zachary A. Correll, Craig R. Dubler, Zachary K. Gotsch
The Pennsylvania State University – *CMPSC 422; Artificial Intelligence*







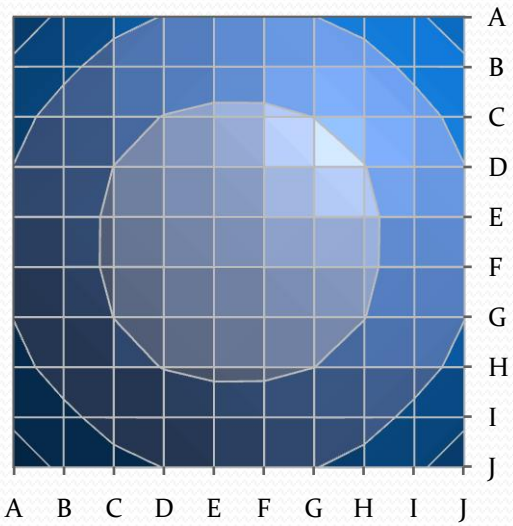
PENNSTATE
The **Computer Science** and **Engineering Department**

Solution Representation

- Divided to three components that have different AI approaches for solution searching:
  - **Ship placement** – Sequential Monte Carlo Method for enemy shot pattern recognition
  - **Targeting** – Genetic algorithm to track enemy ship placement via pattern recognition
  - **Sinking strategy** – Genetic programming to optimized sinking logic

Ship Placement

- Initial placement is based off of the inverse of the experimental random placement data *(See graph)*
- Save all enemy shots into density a map over time
- Apply Monte Carlo algorithm, and place ship to safety intelligently

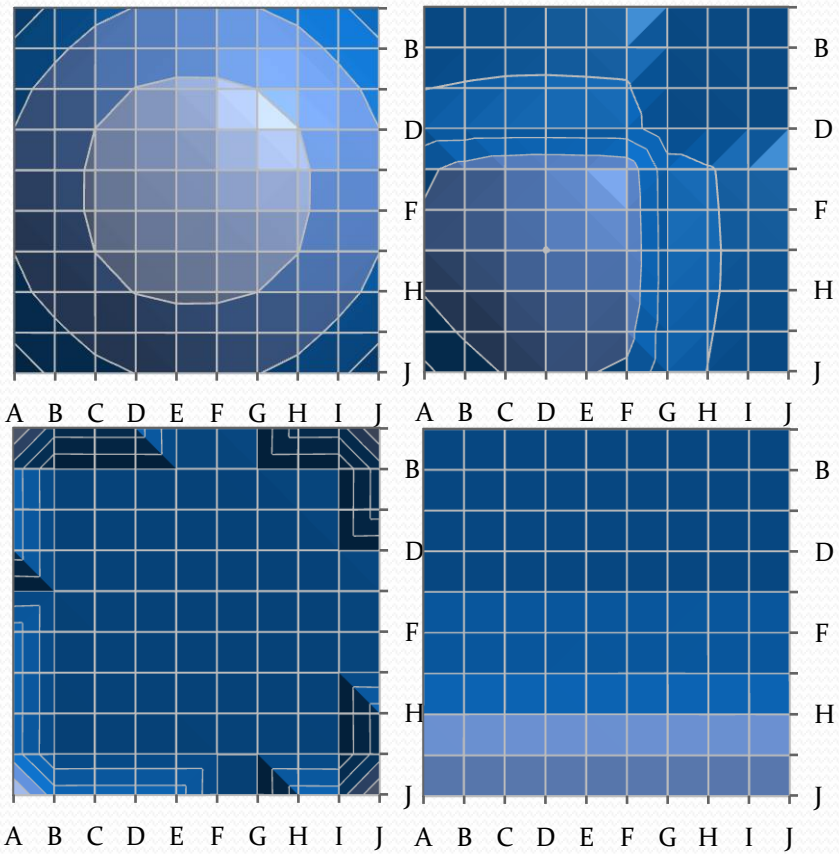


Inverse of random ship placement densities over 1,000,000 rounds

Ship Targeting & Shooting

- Uses a **genetic algorithm** for quick adaptation to enemy ship placement patterns / algorithm, even with random placement
- Defines a gene as five **harmonics** that are randomly chosen at first
- This harmonic represents a **probability distribution** of the likelyhood of enemy ship placement
- **Gene-mutation and cross-over** will converge the gene to a close-fit solution
- **Gene fitness** is done through a Fast Fourier Transformation

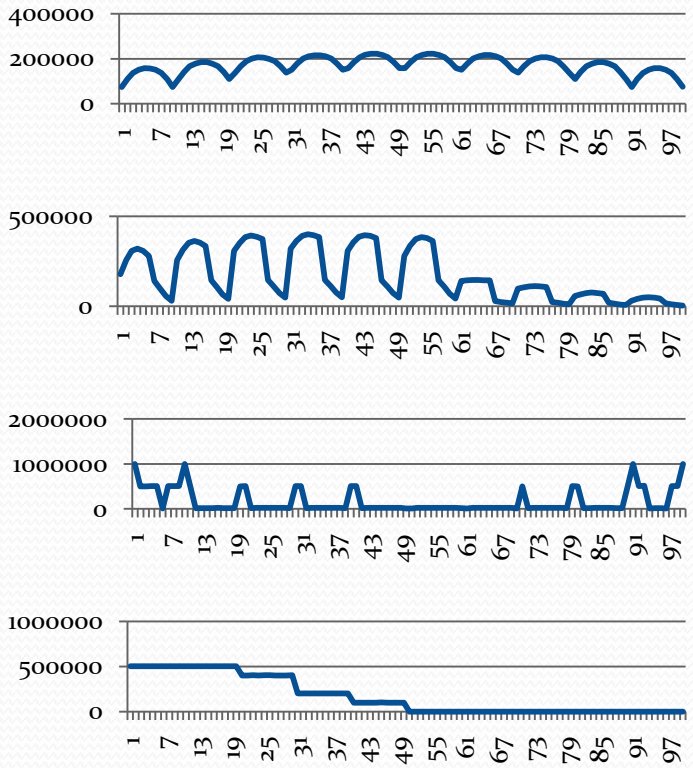
Ship Targeting & Shooting



Placement Density Maps

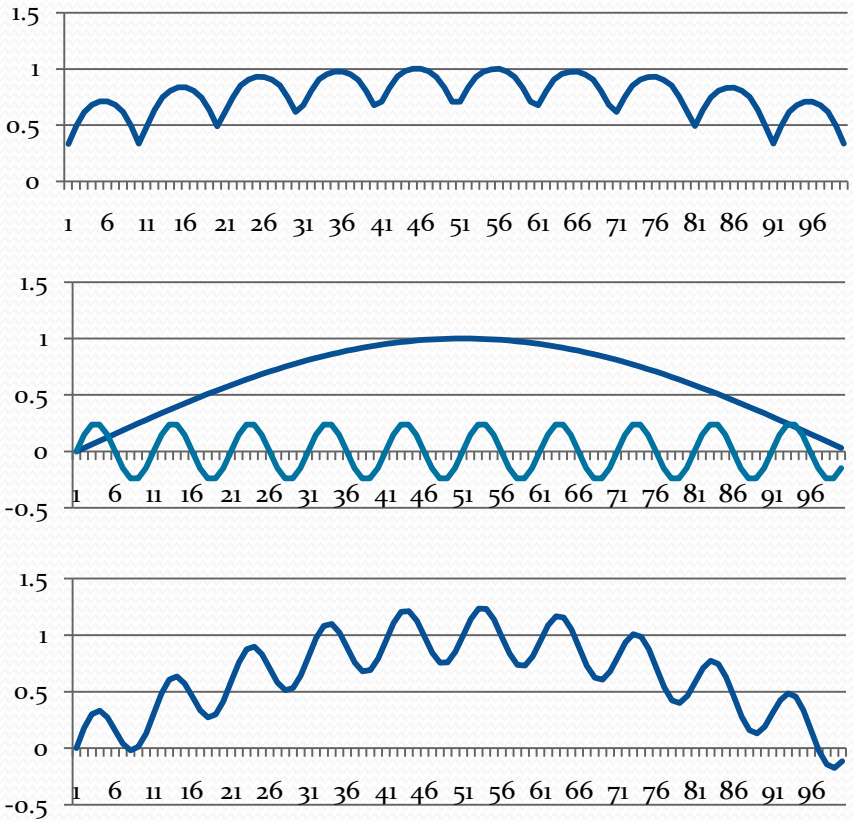
Different methods of ship placement over 1,000,000 rounds. Lighter color indicates higher density of ship placement. From top left, clockwise: Random, lower-left corner favored, bottom wall, one ship in each corner.

Ship Targeting & Shooting



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



Sample Target Data and Gene Harmonics

***Top**: Linear function of the random ship placement density map.*

***Middle**: Harmonic gene with only two components.*

***Bottom**: Successful gene outcome, matching the top pattern.*

Ship Sinking

- Sinking based off of **genetic programming**
- A **state machine** is needed due to the complexity of sinking a ship: Targeting, locking, sinking
- Each **gene** is a series of operations paired with a register machine based on 16 defined instructions
- The **initial** gene pool is pre-trained against hard-coded logic but quickly changes per opponent

Ship Sinking

Register Machine

- **TargetPos**, The current target location, defaults to (0, 0)
- **TempPos**, Temporary position, defaults to (0, 0)
- **TargetDir**, The current target direction, defaults to North / up
- **TargetHit**, Boolean where true if the last shot was successful in hitting a ship
- **TempHit**, Temporary boolean flag for internal usage

Ship Sinking

Sample sinking logic:

1. **MoveFwd** // Keep walking through it
2. **Shoot** // Fire a shot
3. **IfMiss** // If we went past the end
4. **IfTrue** // And we have already sunk the other side..
5. **Jump** // Go back to seeking
6. **IfMiss** // If we went past the end
7. **OppDir** // Flip directions
8. **IfMiss** // Same logic as above
9. **LoadPos** // Load position into register

Implementation & Analysis

- In 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 matching
- Ship sinking logic outperformed hard-coded solutions

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

Questions?

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.