The Journal of Open Source Software

GIANT: General Intelligent AgeNt Trainer

Marko Šmid [1*] **and Miha Ravber** [1*]

1 University of Maribor, Faculty of Electrical Engineering and Computer Science * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

General Intelligent AgeNt Trainer (GIANT) is a versatile, open-source platform designed to train intelligent agents using diverse machine learning techniques and evaluate them in game engine-based environments. Its highly modular architecture supports a wide range of optimization methods, with the current implementation leveraging Genetic Programming (GP) (Koza & Poli, 2005) within the EARS framework ("EARS - Evolutionary Algorithms Rating System (Github)," 2019). GIANT serves as a standardized benchmark for evaluating and comparing solutions in both single-agent and multi-agent environments. The platform integrates concepts of self-organization to enable adaptive evaluation while providing the flexibility to incorporate custom environments and methodologies.

Statement of need

Numerous frameworks exist for training intelligent agents using a variety of machine learning techniques and evaluation environments(Bellemare et al., 2013; Juliani et al., 2020; "Overview of Software Agent Platforms Available in 2023," 2023; Partlan et al., 2022; Song et al., 2020; Towers et al., 2024). However, fairly comparing these solutions remains a persistent challenge. Variations in human implementation, framework performance, and architectural design often introduce inconsistencies that hinder objective analysis (Caton & Haas, 2024; Nguyen et al., 2019). Moreover, when diverse AI controller types must be evaluated within a shared problem domain, most existing frameworks lack the flexibility to support such comparisons under unified conditions.

The importance of robust benchmarking methodologies is well-documented in the literature, with calls for standardized experimental protocols to enhance reproducibility and fairness in optimization research (Bartz-Beielstein et al., 2020; LaTorre et al., 2020). To address these gaps, GIANT offers a modular, standardized platform for training and evaluation across a range of optimization methods. By ensuring consistent experimental conditions, GIANT enables reproducible, cross-method benchmarks tailored to the needs of AI researchers and developers. The platform supports deterministic execution with a fixed seed for repeatable experiments while also allowing dynamic environments to test agent adaptability under varying conditions.

Architecture

The conceptual design of the platform consists of three core components: the Machine Learning Framework, the Evaluation Environment, and the Web App Interface (see Figure 1). These components work together to provide a flexible and extensible platform for training and evaluating intelligent agents.

The Machine Learning Framework includes a collection of optimization algorithms designed for single-agent or multi-agent systems. This framework provides the foundation for training

agents using diverse machine learning techniques, ensuring adaptability across different problem domains.

The Evaluation Environment contains a set of problem domains that simulate real-world challenges. These environments serve as standardized benchmarks for evaluating agent performance, allowing for fair and deterministic comparisons.

The Web API Interface acts as the central integration layer, facilitating communication between the Machine Learning Framework and the Evaluation Environment. Its primary function is to convert individual solutions into a standardized format that can be interpreted and evaluated across different environments.

The modular architecture of GIANT ensures that components remain independent, allowing for the seamless integration or replacement of individual modules without affecting the performance of the overall system. When introducing a new component, only the Web App Interface needs to be adapted to align with its specifications, making the platform highly scalable and adaptable.

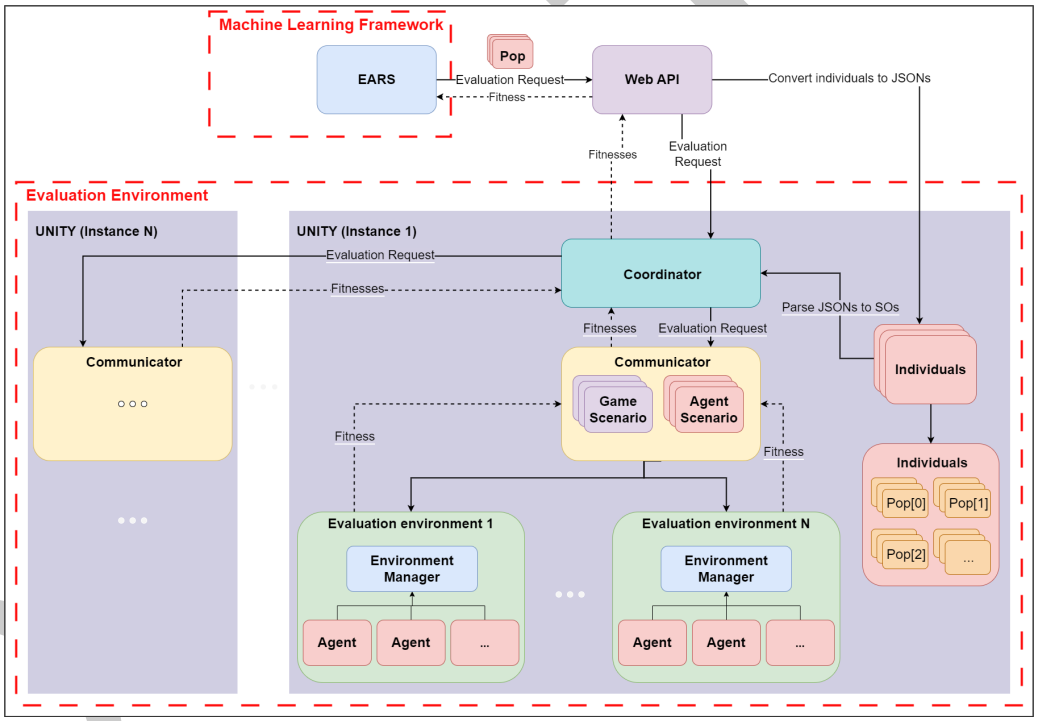

Figure 1: Architecture of GIANT platform.

Included problem domains

The platform includes a set of predefined problem domains, ranging from single-agent to multi-agent scenarios, to facilitate the development of custom problem domains. These problem domains are structured as game-based environments where agents operate under the control of AI-driven decision-making models.

The platform currently supports the following problem domains: Collector, RoboStrike, Soccer, and BombClash. Within these environments, agents can be controlled using a variety of AI controllers, including custom scripts, Behavior Trees (BTs) (Isla, 2022), and Artificial Neural Networks (ANNs) (Richards et al., 1998). Additionally, the platform features a Manual Controller, which allows for direct user input by functioning as a custom script that processes player commands.

- Collector (see Figure 2) is a single-agent environment inspired by ML-Agents' Food Collector (Juliani et al., 2020), where agents navigate a map to collect scattered resources while avoiding obstacles. The goal is to optimize movement efficiency and maximize resource collection within a given time frame.

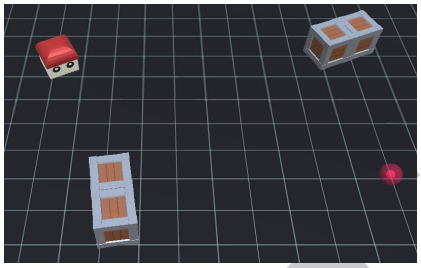

Figure 2: Collector problem domain.

- RoboStrike (see Figure 3) is a multi-agent combat simulation inspired by Robocode (Nelson, 2000), where agents are represented as tanks and placed in a strategic battle arena. Agents must navigate obstacles, track opponents, and employ efficient attack and evasion strategies to survive and eliminate opponents.

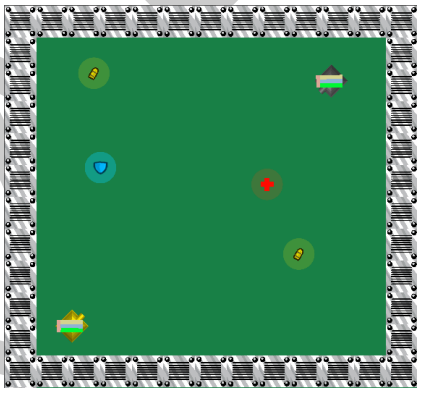

Figure 3: RoboStrike problem domain.

- Soccer (see Figure 4) is a replica of ML-Agents' Soccer Twos (Juliani et al., 2020), a multi-agent environment where agents compete to score goals by strategically positioning themselves, passing, and shooting.

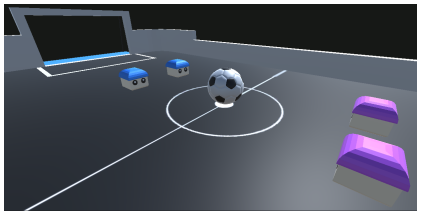

Figure 4: Soccer problem domain.

- BombClash (see Figure 5) is a multi-agent environment inspired by the Bomberland multi-agent AI competition (One, 2021), where agents strategically place bombs to destroy obstacles, eliminate opponents, and evade explosions. Success requires careful planning, opponent prediction, and tactical movement.

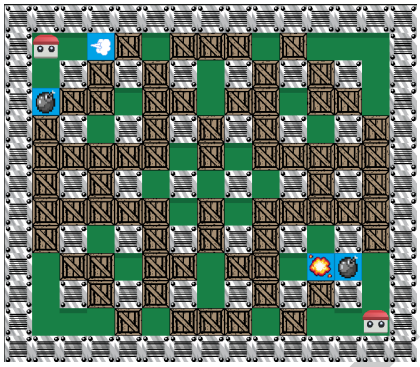

Figure 5: BombClash problem domain.

The platform's flexible control system enables both fully autonomous AI-driven gameplay and human-in-the-loop experimentation, making the platform suitable for a wide range of research and development applications in machine learning, artificial intelligence, and game AI.

Optimization process

The optimization process starts in the EARS framework, which generates an initial population of solutions. This population is sent via a web API to the Unity-based evaluation environment. The API converts individuals into a compatible format and forwards them to the coordinator, Unity's main HTTP server.

The coordinator distributes individuals across available Unity instances based on configuration settings. Each instance, managed by a communicator (an internal HTTP server), conducts evaluations including agents that are controlled by a single controller (e.g., Behavior Trees, Neural Networks, or Manual Controllers). The environment controller updates agents at fixed intervals, ensuring consistent evaluation.

Once the evaluations are completed, the coordinator collects and sends the results back through the web API to EARS, where the optimization continues.

Scalability and Performance Optimization

Since the evaluation phase is the most time-consuming part of the optimization process (Li et al., 2022), the platform implements two levels of parallelization to enhance efficiency. The first level of parallelization operates within a single Unity instance, where multiple simulations are evaluated simultaneously within the same environment but on isolated simulation spaces, ensuring fair and independent evaluation. The second level involves running multiple Unity instances. The number of instances that can run concurrently is constrained only by the processing power of the machine executing the experiment.

To further improve the performance, GIANT provides render toggling, allowing users to disable visual rendering during evaluations. This reduces GPU load and maximizes computational efficiency, especially when graphical output is unnecessary. Additionally, the platform supports increasing the time scale, which accelerates physics simulations by running the simulation at a higher speed than real-time. This feature ensures faster data collection and shortens training cycles without affecting simulation fidelity.

References

Bartz-Beielstein, T., Doerr, C., Berg, D. van den, Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., La Cava, W., Lopez-Ibanez, M., & others. (2020). Benchmarking in optimization: Best practice and open issues. *arXiv Preprint arXiv:2007.03488*.

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, *47*, 253–279. https://doi.org/10.1613/jair.3912

Caton, S., & Haas, C. (2024). Fairness in machine learning: A survey. *ACM Comput. Surv.*, *56*(7). https://doi.org/10.1145/3616865

EARS - evolutionary algorithms rating system (github). (2019). In *GitHub repository*. GitHub. https://github.com/UM-LPM/EARS

Isla, D. (2022). Handling complexity in the halo 2 AI, 2005. *URL: Http://Www. Gamasutra. Com/Gdc2005/Features/20050311/Isla _01. Shtml [21.1. 2010]*.

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., & Lange, D. (2020). Unity: A general platform for intelligent agents. *arXiv Preprint arXiv:1809.02627*. https://arxiv.org/pdf/1809.02627.pdf

Koza, J. R., & Poli, R. (2005). *Genetic programming* (E. K. Burke & G. Kendall, Eds.; pp. 127–164). Springer US. https://doi.org/10.1007/0-387-28356-0_5

LaTorre, A., Molina, D., Osaba, E., Del Ser, J., & Herrera, F. (2020). Fairness in bio-inspired optimization research: A prescription of methodological guidelines for comparing meta-heuristics. *arXiv Preprint arXiv:2004.09969*.

Li, J.-Y., Zhan, Z.-H., & Zhang, J. (2022). Evolutionary computation for expensive optimization: A survey. *Machine Intelligence Research*, *19*(1), 3–23. https://doi.org/10.1007/s11633-022-1317-4

Nelson, M. A. (2000). Robocode. In *GitHub repository*. GitHub. https://github.com/robocode-dev/tank-royale

Nguyen, G., Dlugolinsky, S., Bobák, M., Tran, V., López García, Á., Heredia, I., Malík, P., & Hluchý, L. (2019). Machine learning and deep learning frameworks and libraries for large-scale data mining: A survey. *Artificial Intelligence Review*, *52*(1), 77–124. https://doi.org/10.1007/s10462-018-09679-z

One, C. (2021). Bomberland AI challenge. In *GitHub repository*. GitHub. https://github.com/CoderOneHQ/bomberland

Overview of software agent platforms available in 2023. (2023). *Information*, *14*(6). https://doi.org/10.3390/info14060348

Partlan, N., Soto, L., Howe, J., Shrivastava, S., El-Nasr, M. S., & Marsella, S. (2022). *EvolvingBehavior: Towards co-creative evolution of behavior trees for game NPCs*. https://arxiv.org/abs/2209.01020

Richards, N., Moriarty, D. E., & Miikkulainen, R. (1998). Evolving neural networks to play go. *Applied Intelligence*, *8*(1), 85–96. https://doi.org/10.1023/A:1008224732364

Song, Y., Wojcicki, A., Lukasiewicz, T., Wang, J., Aryan, A., Xu, Z., Xu, M., Ding, Z., & Wu, L. (2020). Arena: A general evaluation platform and building toolkit for multi-agent intelligence. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(05), 7253–7260. https://doi.org/10.1609/aaai.v34i05.6216

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., & others. (2024). Gymnasium: A standard interface for reinforcement learning environments. *arXiv Preprint arXiv:2407.17032*.