Robo-maze Blast Report

Dogà Sara
Widmann Lukas
Askar Sami

Abstract

Bomberman + Evolutionary algorithmens + Tournament Arc goes brr

ACM Reference Format:
Dogà Sara, Widmann Lukas, and Askar Sami. 2025. Robo-maze Blast Report. In . ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

1.1 The game

Robo Maze Blast, created in 2008 by Kai Ritterbusch and Christian Lins, is a clone of the Bomberman game. Also known as Dynablaster, it is a strategy maze-based video game franchise originally developed by Hudson Soft in 1985.

The general goal of Bomberman is to complete the levels by strategically placing bombs in order to kill enemies and destroy blocks. Some blocks in the path can be destroyed by placing bombs near it, and as the bombs detonate, they will create a burst of vertical and horizontal lines of flames. Except for indestructible blocks, contact with the blast will destroy anything on the screen.

1.2 Our Goal

The aim of our project is to explore the efficiency of different Genetic Algorithms to develop 3 agents with strategic competence in the Robo Maze Blast scope, and to compare them by making the agents fight against each other and observe which agent outlives the others more frequently.

2 Background

2.1 Genetic Algorithms

Genetic Algorithms (GA) are optimization algorithms inspired by the process of natural selection and biological evolution. They are widely used to solve complex optimization and search problems in various domains. Due to constrained optimization (e.g., state/action of the game), Genetic Algorithms are a perfect choice for this task [?]: *"Genetic Algorithms (GAs) were selected for their ability to handle complex combinatorial optimization problems [...] and to encode domain-specific constraints"* (Section 1).

The core steps of a typical genetic algorithm can be described as follows:

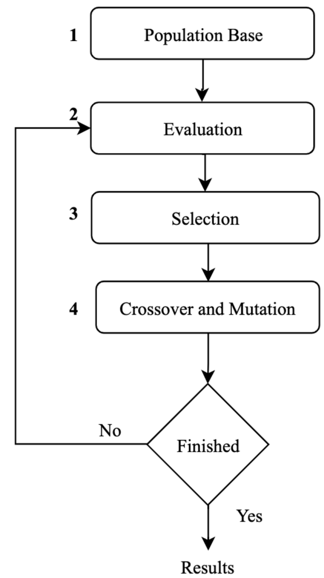

Figure 1: A diagram on the steps of a genetic algorithm

- **Population Base**: Initialize a population from valid chromosomes, i.e. a set of strings that encodes any possible solution. Usually, the initial population is chosen randomly.
- **Evaluation**: Each population solution is evaluated on the basis of a predetermined fitness function.
- **Selection**: Reproductive opportunities are allocated to the chromosomes that represent a better solution to the target problem, and such solutions are selected to form a 'mating pool' for the next generation.
- **Crossover and Mutation**: The selected individuals are then combined to produce offspring by exchanging genetic material. Sometimes small changes can happen in the genetic material, such as bit flips. All of this ensures good exploration of the solution space and diversity.

These steps are repeated for a number of times until an ending criterion is reached.

2.2 Robo Maze Blast's Default AI Agent

3 Agent 1

Agent 1

3.1 Differential Evolution

Simply use the section and subsection commands, as in this example document! With Overleaf, all the formatting and numbering is handled automatically according to the template you've chosen. If

Conference'17, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

you're using the Visual Editor, you can also create new section and subsections via the buttons in the editor toolbar.

First you have to upload the image file from your computer using the upload link in the file-tree menu. Then use the includegraphics command to include it in your document. Use the figure environment and the caption command to add a number and a caption to your figure. See the code for Figure **??** in this section for an example.

Note that your figure will automatically be placed in the most appropriate place for it, given the surrounding text and taking into account other figures or tables that may be close by. You can find out more about adding images to your documents in this help article on including images on Overleaf.

4 Supervised Learning with Jenetics

4.1 Introduction

My objective was to create an AI player based on a GA using human game-play data. For this, I chose the Jenetics library.

4.2 Jenetics

Jenetics is an open-source Java library that provides a genetic Algorithm (GA) framework for solving optimization problems. It abstracts biological evolution principles, such as selection, crossover, and mutation, into reusable software components, enabling users to evolve solutions without implementing a GA from scratch.

4.3 Jenetics Library Key Features:

- **Evolutionary Engine**: Automatically evolves over generations via Darwinian principles
- **Domain-Agnostic Design**: Works with any optimization problem (e.g. game strategy in this case)
- **Built-in Operators**: Selection (e.g. Tournament), Crossover (e.g. Single-point), Mutation (e.g. Gaussian)

For details of the implementation, refer to Jenetics User Manual [Wil24].

4.4 Implementing Gameplay Recording: Code Changes for Data Collection

To implement supervised learning via Jenetics, high quality data is needed. A simple and intuitive appraoch to gather this data is to record a human players actions and the state of the game to gather data. Due to this modification to the player class was needed. Due to this the recordable Player class which is an extension to the player class is born. The key changes to the player class are summarized in 1 the pseudo code. The changes include:

- **Recording trigger**: Modified movement/bomb methods to log actions
- **State capture**: Added position normalization and bomb status flags
- **Data serialization**: Output CSV format for GA compatibility

4.5 How to add Citations and a References List

You can simply upload a `.bib` file containing your BibTeX entries, created with a tool such as JabRef. You can then cite entries from it, like this: [Gre93]. Just remember to specify a bibliography style, as well as the filename of the `.bib`. You can find a video tutorial here to learn more about BibTeX.

Algorithm 2: RecordablePlayer Extension

Class RecordablePlayer **extends** Player
New Data: recordings = [] `// State-action log`
isRecording = false `// Recording toggle`
Key Changes:
move(dx, dy): super.move(dx, dy)
if *isRecording* **then**
  mapToAction(dx, dy) `// Records movement`
placeBomb(): super.placeBomb()
if *isRecording* **then**
  recordAction(BOMB) `// Records bomb placement`
State Capture (New): state ← [normX, normY, bombAvail, bombNear] `// Normalized features`
record ← state + action `// Concatenates state & action`
recordings.add(record) `// Appends to log`
Output (New): `saveRecordings` (filename) → Export recordings as CSV

If you have an upgraded account, you can also import your Mendeley or Zotero library directly as a `.bib` file, via the upload menu in the file-tree.

4.6 Good luck!

We hope you find Overleaf useful, and do take a look at our help library for more tutorials and user guides! Please also let us know if you have any feedback using the **Contact us** link at the bottom of the Overleaf menu — or use the contact form at https://www.overleaf.com/contact.

References

[Gre93] George D. Greenwade. The Comprehensive Tex Archive Network (CTAN). *TUGBoat*, 14(3):342–351, 1993.

[Wil24] Franz Wilhelmstötter. *Jenetics User Manual (Version 8.2.0)*, 2024. Accessed: 07.08.2025.